LA PRATICA DELL'APPLE

Nicole Bréaud - Pouliquen

GRUPPO EDITORIALE JACKSON

La grafica
Il sistema Apple
Il BASIC Applesoft

LA PRATICA DELL'APPLE

La grafica
Il sistema Apple
Il BASIC Applesoft

Nicole Bréaud - Pouliquen

GRUPPO
EDITORIALE
JACKSON
Via Rosellini, 12
20124 Milano

Nicole Breaud-Pouliquen è un ingegnere consulente in informatica "personale". In questo ruolo insegna programmazione da parecchi anni. Pouliquen utilizza l'Apple II da quando questo calcolatore è stato commercializzato in Francia e s'interessa, più specificatamente, dell'utilizzo dei personal computer nell'insegnamento.

L'autore ringrazia per il prezioso lavoro svolto nella stesura dell'edizione italiana la signora Francesca di Fiore e l'ing. Roberto Pancaldi.

Traduzione a cura di Piero Dell'Orco.

Stampato in Italia da:
S.p.A. Alberto Matarelli - Milano - Stabilimento Grafico

SOMMARIO

INTRODUZIONE

Cominciate ad utilizzare il vostro Apple per gradi, e vi renderete conto di quali sono le sue risorse senza spiacevoli inconvenienti.

E' ben noto il linguaggio Basic nelle sue tradizionali funzioni e sono sovente utilizzate istruzioni del tipo "paracadute" che sembrano spesso misteriose. Dedicando un pò di tempo a questo problema sarà possibile comprendere la ragion d'essere di queste istruzioni.

Questa andatura, che il libro pone come obbiettivo, permetterà di riscoprire le possibilità dell'Apple II. Si potranno scrivere programmi con cognizione di causa; sarà possibile spiegare meglio ad amici neofiti il significato di Apple ed avvicinarli al computer con spirito scientifico ed analitico.

Il libro comprende tre capitoli, Il Sistema Apple II, il linguaggio Applesoft, i disegni e la grafica. Gli argomenti sono trattati con il preciso intento di far prendere coscienza delle reali possibilità dello strumento (numeri reali, stringhe di caratteri, loro codifica interna, loro occupazione in memoria etc.).

Si consiglia di effettuare una prima lettura superficiale, al fine di selezionare gli argomenti che interessano in modo specifico e di elaborare un metodo per poi assimilarli nel modo più corretto.

CAPITOLO 1

IL SISTEMA APPLE II

L'Apple II è costruito attorno al microprocessore 6502 che fornisce, insieme ad altre possibilità, il mezzo di indirizzare direttamente fino a 65.536 (64K) parole di 8 bit (1 byte) in memoria, dove una parte è riservata alle funzioni di entrata (Input) e di uscita (Output).

Nei 60 Kbyte rimanenti, il sistema Apple II può disporre di memoria ROM (Read-Only Memory) e di memoria RAM (Random Access Memory).

Per comunicare con il mondo esterno, ad esempio per mezzo di una tastiera, di un video, di un registratore a cassette, il sistema contiene delle specifiche interfacce integrate nella scheda principale, ma è possibile la connessione di schede di interfaccia supplementari (8 al massimo) che aumentano le possibilità di comunicazione e memorizzazione; (per esempio le schede di interfaccia possono controllare le unità floppy-disk, stampanti, modem etc.).

Il software di base del sistema Apple II risiede nella memoria ROM (12 Kbyte) e permette, all'accensione del sistema, il dialogo sia con il Monitor che con l'interprete BASIC.

L'utilizzatore può lavorare sul sistema in modo diretto e interattivo, in linguaggio evoluto, in linguaggio macchina oppure utilizzando programmi precedentemente caricati in memoria RAM.

L'HARDWARE

Si descrivono le caratteristiche principali del microprocessore, i due tipi di memoria utilizzata e le capacità del sistema di I/O implementato sulla scheda principale (Mainboard).

IL MICROPROCESSORE 6502

Il circuito integrato del tipo 6502 a 28 piedini è fabbricato dalla M.O.S. Technology Inc., dalla Synertek e dalla Rockwell.

L'unità di calcolo, che gestisce registri a 8 bit in parallelo, è capace di ricevere o di canalizzare informazioni su 8 bit che vanno o vengono ad una delle 65536 posizioni di memoria indirizzabili (gli indirizzi sono forniti con parole di 16 bit 2 byte).

Un registro accumulatore, due registri indice, un registro puntatore di catasta (256 byte di profondità massima), un registro di stato del processore ed un contatore (16 bit), associati a 56 istruzioni e 13 modi d'indirizzamento, costituiscono i mezzi di programmazione del microprocessore.

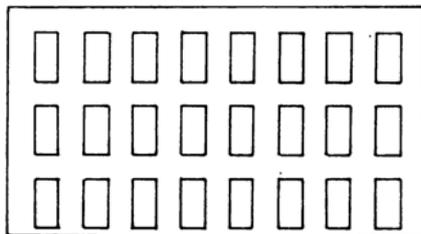
La cadenza delle operazioni è regolata da un clock che scandisce il tempo ad una velocità di 1.023.000 cicli al secondo, cioè 1,023 MHz; il microprocessore esegue circa 500.000 addizioni o sottrazioni al secondo. Un sistema di interruzione (interrupt) permette di prendere in considerazione segnali esterni e di sospendere l'esecuzione delle istruzioni in corso.

LE MEMORIE RAM

Sono del tipo dinamico costruite secondo la tecnologia NMOS (moderato consumo, elevato livello d'integrazione) e necessitano dunque di un rinfresco automatico.

Questa 'rigenerazione' periodica è realizzata ad ogni ciclo del microprocessore. Alcune zone della memoria RAM sono utilizzate come memoria dello schermo video (visualizzazione di testi o di immagini grafiche) e ovviamente sfruttano il rinfresco del sistema per ottenere una visualizzazione sufficientemente persistente.

I circuiti integrati che compongono la memoria RAM sono disposti in linea (da 1 a 3) in ragione di 8 circuiti di memoria per linea. Ogni circuito integrato o 'chip' di memoria può contenere:

- 4096 bit (IC Mostek 4096)
- 16384 bit (IC Mostek 4116)

Le tre file con gli 8 zoccoli per i circuiti integrati di memoria RAM.

Su di una stessa fila, tutti i 'chip' devono contenere il medesimo numero di bit. Per esempio, una linea di 8 chip da 16384 bit costituisce una memoria da 16384 byte.

Si potranno comporre 9 differenti configurazioni di memoria RAM, da 4 Kbyte (8 IC '4096'), fino a 48 Kbyte (24 IC '4116') con valori intermedi di 32 Kbyte (16 IC '4116') o per 24 Kbyte (16 IC '4096' e 8 IC '4116').

Il tempo di accesso ad una qualunque posizione della memoria RAM è di 250 nano-secondi (10^{-9} sec.), e il tempo di ciclo di 375 nano-secondi.

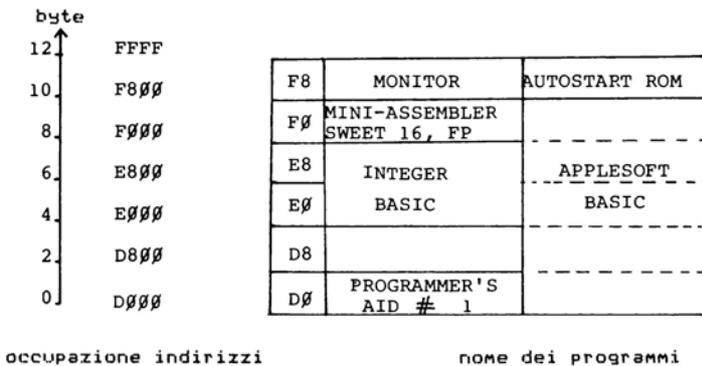
LE MEMORIE ROM

Le 6 posizioni situate tra il microprocessore e la memoria RAM sono riservate a blocchi da 2 Kbyte di memoria ROM (circuiti integrati a 24 piedini del tipo 9316B).

Un certo numero di programmi sono disponibili in ROM, come il Monitor, l'Integer BASIC, l'Applesoft BASIC ed altre 'utility' di sistema.

Queste memorie hanno gli indirizzi piazzati nella parte più alta del sistema ed occupano 12 Kbyte di spazio totale. Ecco come sono situati i programmi in ROM:

byte				
12	FFFF			
10	F8ØØ	F8	MONITOR	AUTOSTART ROM
8	FØØØ	FØ	MINI-ASSEMBLER SWEET 16, FP	
6	E8ØØ	E8	INTEGER	APPLESOFT
4	EØØØ	EØ	BASIC	BASIC
2	D8ØØ	D8		
0	DØØØ	DØ	PROGRAMMER'S AID # 1	

occupazione indirizzi — nome dei programmi

Ciascuno dei programmi ha dunque un indirizzo ben preciso, in particolare l'uno o l'altro dei programmi Monitor o Autostart dev'essere nella parte alta della memoria, poichè all'avviamento, il microprocessore, eseguirà automaticamente un programma il cui indirizzo iniziale è posizionato nella parte alta.

Se l'interprete Applesoft è installato in ROM, impedisce l'utilizzo di altri programmi 'utility' in ROM (per esempio il mini-assembler).

ESTENSIONE DELLA MEMORIA

Una particolarità molto interessante del sistema Apple è che, per mezzo di semplice commutazione, (programmata o manuale), si possono decodificare i blocchi di ROM installati sulla scheda principale. Nella decodifica del sistema si libera uno spazio di indirizzi di 12 Kbyte che sarà disponibile per altri programmi in ROM installati su una scheda di interfaccia chiamata Apple Firmware Card, o per altri byte di memoria RAM (16 Kbyte) installati su un'altra scheda di interfaccia chiamata Apple Language Card.

La scheda Apple Firmware sostituirà le ROM della scheda principale con le proprie ROM (per esempio l'interprete Applesoft in ROM); grazie ad una commutazione 'scheda principale-scheda firmware' che si ottiene:

sia con un comando da programma, facendo riferimento agli indirizzi riservati a questo scopo:

$C080 (scheda Firmware collegata)
$C081 (scheda principale collegata).

sia per posizionamento di un commutatore a due posizioni posto sulla scheda firmware:

in posizione alta (scheda firmware)
in posizione bassa (scheda principale).

Questi due sistemi di commutazione sono utilizzabili indifferentemente.

Per esempio, se il commutatore è in posizione alta all'accensione del sistema e se la scheda firmware contiene l'Applesoft, l'inizializzazione seguita da RESET e da CTRL-B pone il sistema in Applesoft e sullo schermo si avrà il simbolo di "prompt" dell'Applesoft.

Se la scheda principale contiene il mini-assembler in ROM e lo si vuole, in seguito, utilizzare, sarà necessario premere RESET poi C081 e RETURN per disattivare la scheda firmware ed avere libero accesso a questo programma in ROM posto sulla scheda principale (con F666 G).

La scheda Apple Language sostituirà le ROM della scheda principale con le proprie RAM nello spazio di indirizzo compreso tra $D000 e $FFFF.

Questo sistema permette, in alcuni casi, di portare la capacità di memoria disponibile per l'utilizzatore a 64 Kbyte di RAM.

Per mezzo di un comando programmato (indirizzi riservati C08X), si chiederà al sistema Apple di tener conto sia dello spazio da $D000 a $FFFF, sia della memoria ROM della scheda principale (ad eccezione del Monitor che è quello situato sulla scheda Language), sia della memoria RAM installata sulla scheda Apple Language (16 Kbyte). Nel caso della selezione della memoria RAM, il conseguimento di 16 Kbyte si ottiene dalla selezione degli indirizzi da $E000 a $FFFF da una parte che procura 8 Kbyte, e dalla commutazione (o swapping) di uno dei due blocchi RAM di 4 Kbyte sullo spazio da $D000 a $DFFF.

In effetti, lo spazio da $C000 a $CFFF non è utilizzabile, poichè in questa zona sono implementate le funzioni di I/O.

In ogni caso l'accesso ai 16 Kbyte di memoria RAM addizionali non è automatico quando viene utilizzato il BASIC, poichè la protezione in scrittura è in funzione su quelle memorie RAM che contengono il linguaggio BASIC.

Sono possibili e programmabili le seguenti combinazioni:

- lettura e scrittura della RAM protetta
- lettura ROM e scrittura RAM autorizzata
- lettura ROM e scrittura RAM protetta
- lettura e scrittura RAM autorizzata.

ENTRATA/USCITA (I/O)

Nello spazio dall'indirizzo $C000 a $CFFF, sono disposti 4096 byte riservati esclusivamente alle funzioni di I/O delle informazioni.

Le funzioni comunicano con il microprocessore per mezzo delle istruzioni scritte in questi indirizzi, sia la loro implementazione sulla scheda principale o su schede di interfaccia per periferiche.

Sulla scheda principale le funzioni di I/O (Built - in I/O) sono scritte su 128 byte da $C000 a $C07F.

In entrata (Input):

$C000 tastiera | S | Dati ASCII |

i 7 bit di destra rappresentano il codice ASCII del tasto premuto. L'ottavo bit, all'estrema sinistra, è chiamato 'Keyboard Strobe'; quest'ultimo è generato dalla decodifica dei tasti ed assume il valore 1 ogni volta che viene premuto un tasto.

$C060 lettura dell'unità a cassetta | |//////////|

solo il bit più a sinistra è utile e rappresenta il segnale letto sulla cassetta e trasmesso dall'uscita EAR-PHONE fino al connettore marcato IN, posto sul retro di Apple. Questo bit viene testato da un programma del Monitor e decodificato quando è presente l'istruzione LOAD.

$C061, $C062, $C063

Input del tipo tutto-niente (on-off) provenienti dai pulsanti delle "paddle" e trasmesse attraverso il connettore "I/O Game" sui piedini chiamati PB0, PB1 e PB2. Solo il bit più significativo è considerato e vale 1 se il pulsante è premuto, 0 in caso contrario.

$C064, $C065, $C066, $C067

Input analogici provenienti dalle "paddle" e trasmesse attraverso il connettore "I/O Game" sui piedini chiamati GC0, GC1, GC2, GC3.

Ogni "paddle" è costituita da un potenziometro (resistenza variabile) da 150 Kohm alimentato da una tensione costante di 5 Volt e collegata all'entrata di un circuito monostabile che fornisce un impulso di durata proporzionale al valore della resistenza.

Un sottoprogramma del Monitor, chiamato PREAD testa la durata e i cambiamenti degli impulsi e rinvia nel registro Y un valore numerico compreso tra 0 (potenziometro a 0 ohm) e 255 (potenziometro a 150 Kohm).

Il monostabile è inizializzato o resettato da un segnale inviato da un programma che fa riferimento all'indirizzo $C070 (reset del conteggio della durata o PTR/G).

In linguaggio BASIC le entrate analogiche sono testate dalle istruzioni E1 = PDL(1) e E0 = PDL(0). Le variabili E1 e E0 conterranno dei valori compresi tra 0 e 255, proporzionali ai valori di resistenza delle "paddle".

Gli ingressi analogici sono facilmente utilizzabili per misurare le temperature per mezzo di termistori oppure intensità luminose per mezzo di fotoresistenze , tutto ciò è realizzabile con semplici circuiti che sostituiscono le paddle.

In uscita:

$C010 reset del bit "Keyboard strobe".

Questo bit è posizionato all'estrema sinistra dell'indirizzo $C000 ed è posto a 1

per la decodifica della tastiera ogni volta che un tasto viene premuto. Per testare se un nuovo tasto è stato premuto, bisognerà assolutamente rimettere il bit a 0. Il segnale di RESET è inviato facendo riferimento all'indirizzo $C010.

$C020 registrazione su cassetta

Facendo riferimento a questo indirizzo, è generato un segnale passante da 0 mV. a 25 mV. o da 25 mV. a 0 mV. sul connettore OUTPUT dell'Apple che viene trasmesso verso l'entrata MICROPHON della cassetta per la registrazione.

Facendo variare la frequenza e la durata del segnale, un programma del Monitor realizzerà il salvataggio dei dati su cassetta dopo un'adeguata codifica.

$C030 altoparlante

Ogni volta che si farà riferimento a questo indirizzo, l'altoparlante emetterà un leggero "click". Facendo riferimento a questo indirizzo periodicamenete e continuamente, si otterranno suoni di tonalità e di durata variabile.

$C058, $C059, $C05A, $C05B, $C05C, $C05D, $C5E, $C05F

Corrispondono a uscite del tipo tutto-niente (on-off), chiamate "annunciators" disponibili su piedini AN0, AN1, AN3 del connettore "I/O Game".

Ciascuna uscita può controllare un dispositivo esterno (lampadine, LED, relais, altoparlanti, TTY etc..).

Gli indirizzi sono raggruppati in coppia. Indirizzando una posizione pari, l'uscita corrispondente è posta a 0. Indirizzando una posizione dispari, l'uscita corrispondente è posta a 1.

$C040

E' chiamata "utility Strobe" oppure C040Strobe sul connettore I/O Game.

Su questo piedino, si può ottenere, in uscita, un impulso di durata 500 nanosecondi (0,5 microsecondi) facendo riferimento all'indirizzo $C040.

Per realizzare queste differenti funzioni di entrata-uscita, si deve dunque far riferimento agli indirizzi che sono loro riservati. E' sufficiente, in realtà, chiedere al microprocessore di leggere o scrivere questi indirizzi per ottenere un risultato.

Un'operazione di scrittura si realizza attraverso una lettura seguita da una scrittura a livello di microprocessore. Quindi alcune funzioni potranno essere realizzate indiferrentemente con lettura o scrittura, ed altre unicamente con lettura.

SOMMARIO DELLE FUNZIONI DI I/O

Funzioni	Indirizzi decimali	Esadeci-mali	
Altoparlante	49200 - 16336	$C030	lettura
Registrazione Lettura cassette	49184 - 16352 49256 - 16288	$C020 $C060	lettura lettura
Uscite del tipo tutto o niente Annunciators	49240 - 16299 a a 49247 - 16289	$C058 a $C05F	lettura o scrittura
Ingressi del tipo tutto o niente (pulsanti)	49249 - 16287 49250 - 16286 49251 - 16285	$C061 $C062 $C063	lettura
Ingressi analogici (paddle)	49252 - 16284 49253 - 16283 49254 - 16282 49255 - 16281	$C064 $C065 $C066 $C067	lettura
Azzeramento ingressi analogici	49264 - 16272	$C070	lettura o scrittura
Impulso 0.5 microS (utility strobe)	49216 - 16320	$C040	lettura

Il sistema Apple può comunicare con periferiche grazie a speciali circuiti di interfaccia e a programmi di gestione di segnali e comandi particolari di I/O per mezzo degli 8 connettori, destinati a questo scopo, numerati da 0 a 7.

Ogni connettore è dotato di 50 contatti sui quali sono presenti i segnali per gli indirizzi (16 bit), per i dati (8 bit), per la selezione di indirizzi (dunque periferiche), per la generazione di "interrups", per poter disabilitare le ROM della scheda principale etc.

Questa architettura di I/O fa si che le periferiche siano viste dal microprocessore come delle posizioni di memoria alle quali sono applicabili tutte le possibilità di indirizzamento della memoria.

A ogni connettore, salvo il connettore 0, il sistema Apple fa corrispondere 256 indirizzi del proprio spazio in memoria. Questa pagina di 256 posizioni contiene un programma in ROM che realizza le specifiche funzioni di I/O della periferica collegata.

Questa ROM è posta sulla scheda di interfaccia che è, in questo modo, dotata di una certa "intelligenza".

L'inizio di questa pagina ha per indirizzo $Cs00, dove "s" è il numero del connettore dov'è inserita la scheda di interfaccia.

Inoltre, ciascun connettore dispone di 16 posizioni ripartite tra gli indirizzi da $C080 a $C0FF (per l'insieme degli 8 connettori) e ogni scheda di interfaccia può utilizzare quegli indirizzi a proprio modo,per realizzare funzioni di I/O sulla periferica collegata a quella scheda.

Infine, uno spazio da 2048 indirizzi (da $C800 a $CF00) è utilizzabile per tutte quelle periferiche che necessitano di un programma di entrata-uscita più elaborato e sarà posto in ROM sulla scheda di interfaccia corrispondente.

Ogni interfaccia può avere questa ROM da 2 Kbyte, ma solo una alla volta nello stesso momento, potrà essere in "linea" con il sistema Apple.

Per ogni connettore sono riservate delle locazioni di memoria in RAM , memorie localizzate tra $400 e $7FF. Ogni connettore dispone di 8 byte (salvo il connettore 0), utilizzabili come "Scratch-pad" (memorie temporanee).

Queste posizioni si trovano all'interno della pagina TEXT (testo) o GR (bassa risoluzione), ma le funzioni grafiche non interferiscono in alcun modo su di esse.

UTILIZZAZIONE DEI 4K RISERVATI ALL'I/O

numero di byte	Indirizzi	Utilizzi
128	ØCØØØ a ØCØ7F	Controlli di I/O sulla scheda principale
16 x 8	ØCØ8Ø a ØCØFF	16 indirizzi per ciascuno degli 8 connettori di I/O
256 x 7	ØCsØØ a ØCsFF s da 1 a 7	256 indirizzi per la scheda di int. per un programma di I/O in ROM
2Ø48	ØC8ØØ a ØCFFF	2048 indirizzi per altri programmi in ROM sulle schede di interfaccia (una sola scheda da 2 Kbyte puo' essere attiva)

IL SOFTWARE DELL'APPLE II

Data la sua architettura, il sistema Apple offre numerose possibilità per l'implementazione del software in RAM o in ROM.

L'utilizzatore può disporre della quasi totalità dei 48 Kbyte di RAM della scheda principale per il software, e può operare sull'Apple a tutti i livelli (Monitor e linguaggi evoluti).

Il programma Monitor

Il software realizzato in ROM fornito con l'hardware sulla scheda principale, occupa i 2 Kbyte posizionati nella parte più alta della memoria centrale dall'indirizzo $F800 all'indirizzo $FFFF.

Si accede al programma Monitor, con la ROM Autostart installata, con l'istruzione diretta CALL -151 (1).

Il simbolo di "pronto", costituito da un asterisco (*), si presenta sul video davanti al cursore.

Si esce da questo programma richiamandone un altro tra quelli disponibili sul sistema Apple (per esempio l'interprete BASIC), o eseguendo tutti i programmi scritti in linguaggio macchina dal Monitor.

Il Monitor ha una doppia funzione: serve a volte da supervisore, ma anche da "schiavo" al sistema Apple. E' "schiavo", poichè esegue su richiesta le routine di I/O riguardanti la tastiera, il video, le cassette, le paddle; può essere interpellato per esaminare delle locazioni di memoria, trasferire zone di memoria da un indirizzo ad un altro, eseguire programmi passo-passo in linguaggio macchina per poter osservare agevolmente il contenuto dei registri etc.

Ma è anche il supervisore di tutti i componenti del sistema che normalmente se ne servono.

---

(1) *Con le precedenti versioni di Apple il Monitor era presente facendo seguire, all'accensione del sistema, la pressione del tasto "RESET".*

Comandi del Monitor

Esame del contenuto di una locazione di memoria:

*FF0F 'Return'
FF0F - B0

L'indirizzo seguito da Return, fornisce il valore in codice esadecimale (dopo il trattino) del contenuto di questa locazione di memoria.

Esame di un blocco di memoria:

*800.807 'Return'
0800 - FF FF 00 00 FF FF 00 00

L'indirizzo di partenza, un punto, l'indirizzo di fine zona, forniscono i contenuti delle locazioni di memoria successive in ragione di 8 byte per linea visualizzata.

*'Return'
0808 - FF BF 00 00 FF FF 00 00

'Return'permette di proseguire la visualizzazione delle memorie fornendo, di volta in volta, le successive 8 locazioni di memoria.

*.813 'Return'

Un punto seguito dall'indirizzo di fine zona permette l'esame delle locazioni di memoria tra l'ultima visualizzazione e la locazione di memoria specificata.

Modifica del contenuto di una locazione di memoria:

*800 : 00 FF FF

L'indirizzo di una locazione di memoria, due punti, i valori dati a queste locazioni, separate da uno spazio, registrano questi nuovi valori.

* : FF 00

Continua l'operazione di modifica precedente.

Spostamento di una zona di dati:

*400 < F800.FC00 M (MOVE)

L'indirizzo di destinazione, <, l'indirizzo di partenza, un punto, l'indirizzo di fine zona da spostare seguito da M, trasferiscono i contenuti dalla memoria da una zona all'altra.

Verifica del trasferimento:

```
*400 < F800.FC00 V        (VERIFY)
```

Verifica l'operazione precedente e visualizza le locazioni che presentano discordanza.

Scrittura e lettura su cassetta:

```
*400.5FFF R               (READ)
```

L'indirizzo di partenza, un punto, l'indirizzo di fine zona seguita da R, leggono dati dalla cassetta che vengono caricati in memoria nella zona specificata.

```
*4000.5FFF W              (WRITE)
```

Questo comando realizza la registrazione di una zona di memoria specificata su cassetta.

Dissemblatore:

```
* 3D0 L                     (LIST)
3D0      4C   BF   9D   JMP $9DBF
3D3      4C   84   9D   JMP $9D84
.
.
3FB      4C   65   FF   JMP $FF65
```

L'indirizzo di inizio di un programma in linguaggio macchina, seguito da L realizza la decodifica delle istruzioni di programma dal linguaggio esadecimale a linguaggio mnemonico (o miniassemblatore).

Ogni istruzione può necessitare di uno, due o tre byte di memoria.

Il comando L visualizza solo venti istruzioni alla volta.

Esecuzione di un programma in linguaggio macchina:

```
*3D0 G                    (GO)
```

L'indirizzo di inizio del programma seguito da G provoca l'esecuzione del programma per mezzo del microprocessore.

*3D0 T (TRACE)

Parallelamente all'esecuzione sono visualizzati i contenuti dei registri dopo ogni istruzione eseguita.

*3D0 S (STEP)

L'esecuzione è ottenuta passo-passo, istruzione per istruzione, ripremendo il tasto S a ciascun passo.

Lettura e modifica dei registri del microprocessore:

A è il registro accumulatore (8 bit) del 6502 per il quale transitano i dati e i risultati delle operazioni aritmetiche e logiche.

X,Y sono due registri indice (8 bit) che permettono l'indirizzamento indicizzato.

P è il registro di stato del microprocessore (8 bit) che contiene il valore del riporto, del supero di capacità, lo stato nullo dell'accumulatore o lo stato negativo, lo stato "interrupt" del microprocessore, lo stato del modo di calcolo (decimale o binario).

S è il registro puntatore di catasta, contiene l'indirizzo dell'ultimo dato ordinato in cima alla catasta (STACK).

Visualizzazione e modifica dei registri:

```
* 'Ctrl E''Return'
  A=40   X=01   Y=D8   P=30   S=63

* :FF 00 33
* 'Ctrl E'
  A=FF   X=00   Y=33   P=30   S=63
```

Aritmetica esadecimale:

```
* F + E
=  1D

* FE - F
=  EF
```

Un numero esadecimale di due cifre al massimo + o - un numero esadecimale di due cifre al massimo, comanda la visualizzazione del risultato dell'operazione in codice esadecimale.

Sottoprogrammi facenti parte del MONITOR

$FF3F **IOSAVE** salva il contenuto dei registri A-X-Y-P-S negli indirizzi da $45 a $49

$FF4A **IOREST** Ricarica nei registri A-X-Y-D-S i valori salvati agli indirizzi da $45 a $49.

$FD1E **PREAD** Conversione dell'entrata analogica gc (0 a 3) in un valore numerico da 0 a 255. Il registro X è caricato prima della chiamata con il numero gc. Il registro Y è caricato dopo la chiamata, con il valore cercato.

SETCOL, NEXTCOL, PLOT, HLINE, VLINE, CLSCR, CLRTOP, SCRN
Sono sottoprogrammi per il tracciamento di grafici in bassa risoluzione.

$FBDD **BELL, BELL 1** Inviano dei "bip-bip" sull'altoparlante.

$FCA8 **WAIT** E' un sottoprogramma di temporizzazione. Sia "a" il contenuto dell'accumulatore prima della chiamata, in questo caso il tempo di attesa per ciascuna chiamata al sottoprogramma è di:
Dms = (13 + 12,5 × a + 2,5xaxa)/1023

- Qualche valore caratteristico per a e D :

a	Dms
18	1.02
88	20.01
200	100.21
0 o 255	162.04

Al termine, il registro accumulatore è settato a 0 ed il contenuto dei registri X e Y non è modificato.

Sottoprogramma di input dei caratteri

$FD0C **RDKEY** Input di un carattere. L'indirizzo della routine di input dovrà risiedere in $38 e $39 prima della chiamata.

$FD1B	**KEYIN**	Routine di entrata di un carattere proveniente dalla tastiera. Il codice del carattere è inviato in accumulatore.
$FD35	**RDCHAR**	Routine speciale di entrata di caratteri provenienti dalla tastiera preceduti da ESC.

Sottoprogrammi di entrata di linea:

$FD6A	**GETIN**	Registra l'insieme dei caratteri di una linea nella zona buffer di entrata che inizia a $200. X conterrà il numero dei caratteri della linea.
$FD67	**GETLNZ**	Invia un ritorno carrello ('Return') prima di chiamare GETLN.
$FD6F	**GETLN1**	Opera come GETLN senza inviare un carattere identificatore di linguaggio (]<*) prima della registrazione.

Sottoprogrammi di uscita caratteri:

$FDED	**COUT**	Uscita del carattere. Il carattere da inviare è presente nell'accumulatore. L'indirizzo della routine di uscita dovrà figurare in $36 e $37 prima della chiamata.
$FDF0	**COUT1**	Routine di visualizzazione di un carattere sullo schermo con gestione della posizione del cursore.

Questo elenco di sotoprogrammi non è completa, ma fornisce un'idea delle funzioni generate dal programma Monitor.

Comandi che permettono di uscire dal programma Monitor

* 'Ctrl B' 'Return'

]

Attiva l'interprete BASIC in ROM situato sulla scheda principale o sulla scheda firmware.

La selezione verrà eseguita con una commutazione programmata ($C080 o $C081) o manuale agendo sul commutatore posto sulla scheda firmware (interruttore alto o basso).

* 'Ctrl C' 'Return'

]

Riattiva l'interprete BASIC in funzione, senza cancellare il programma in corso.

* F666 G

!

Attiva il mini assemblatore in ROM se la scheda principale è stata selezionata precedentemente e se quest'ultima ne è dotata.

* 'CTRL Y' 'Return'

*

Attiva un programma in linguaggio macchina, scritto dall'utilizzatore, il cui indirizzo di partenza è $3F8.
Si scriverà in $3F8 un'istruzione di salto incondizionato all'inizio del programma utilizzatore.

* 3D0 G

]

Attiva l'interprete BASIC in funzione senza modificare quanto inserito durante il caricamento del DOS in RAM.

IL PROGRAMMA AUTOSTART ROM

Questo software, realizzato in ROM, è fornito con l'hardware APPLE II PLUS sulla scheda principale, e occupa le posizioni più alte della memoria da $F800 a $FFFF.

Al momento dell'accensione il programma "Autostart" è eseguito e attiva l'interprete BASIC selezionato, o meglio, se è presente la scheda di interfaccia per la gestione delle unità a disco magnetico (floppy disk) questa sarà "trovata" dal sistema e, in questo caso, sarà caricato in RAM da disco il DOS (Disk Operating System).

Premendo il tasto RESET, dopo la prima accensione, viene restituito il controllo del sistema all'interprete in funzione e il programma in corso e le relative variabili non vengono modificate.

Il programma Autostart contiene tutte le funzioni e i sottoprogrammi del programma Monitor con l'eccezione dei comandi S e T.

Il programma "Autostart" permette dei comandi di "editing" più interessanti (ESC-I, J, K, M) e il comando di arresto del comando LIST (CTRL-S).

IL MINI ASSEMBLATORE

Questo software realizzato in ROM, è disponibile con l'hardware Apple con l'interprete Integer BASIC in ROM sulla scheda principale.

E' un linguaggio assemblatore delle istruzioni del 6502, ma non permette di dare alle linee di istruzione delle etichette simboliche.

L'accesso a questo linguaggio si ottiene partendo dal programma Monitor con il comando:

```
*F666 G
!
```

Il punto esclamativo indica la presenza in linea dell'assemblatore.
Si esce da questo linguaggio premendo RESET o per mezzo del comando:

```
*F669 G
*
```

che riattiva il programma Monitor.

Esempio di assemblaggio di istruzioni:

```
!300 : LDA # 12 'RETURN'
0300 - A9 12
! JSR $FCA8
0302 - 20 A8 FC
! BRK
0305 - 00
!
```

Un errore nella sintassi viene segnalato con il segno $\wedge$. Uno spazio è necessario

prima di introdurre il codice dell'istruzione, salvo se quest'ultimo è preceduto da : (due punti) e dall'indirizzo assoluto dell'istruzione.

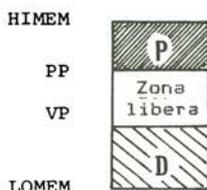
L'INTEGER BASIC

Questo linguaggio, che risiede in ROM, è installato sulla scheda principale o sulla scheda "Firmware" e occupa 4 Kbyte dall'indirizzo $E000 all'indirizzo $EFFF.

Si tratta di un semplice interprete BASIC a variabili intere.

Tratta stringhe di caratteri in modo molto specifico: dette stringhe devono essere dimensionate a priori e ogni stringa costituisce una lista di caratteri.

Le matrici a più dimensioni non sono ammesse.

L'Integer BASIC consente un riconoscimento più rapido degli errori di sintassi e la visualizzazione dei valori delle variabili durante le loro variazioni.

L'implementazione in memoria RAM di un programma interpretato dall'Integer BASIC è realizzata in due zone: la zona delle istruzioni di programma P situata nella parte alta della memoria, la zona delle variabili D del programma situata nella parte bassa della memoria.

Gli indirizzi di inizio e fine di queste due zone sono accessibili ai seguenti indirizzi:

Indirizzi		Contenuto
esadecimali	Decimali	
$4A,$4B	74,75	Indirizzo di inizio della zona di dati o LOMEM
$4C,$4D	76,77	Indirizzo di inizio della zona del programma o HIMEM
$CA,$CB	202,203	Indirizzo di inizio della zona del programma o PROGRAM COUNTER
$CC,$CD	204,205	Indirizzo di fine della zona dei dati o VARIABLE POINTER

Il linguaggio BASIC Applesoft

Questo linguaggio, che risiede in ROM è installato sulla scheda principale o sulla scheda firmware e occupa 10 Kbyte dall'indirizzo $D000 fino a $F7FF.

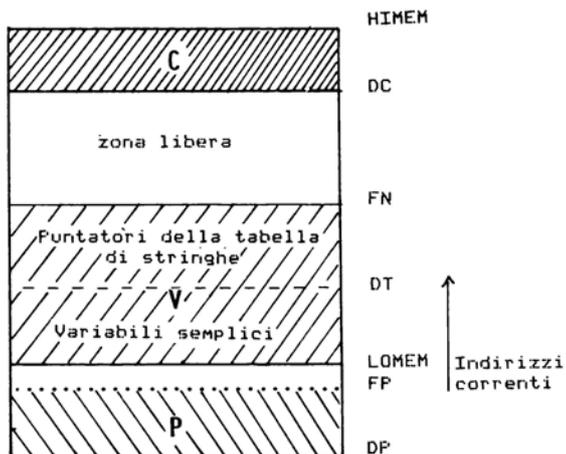
Si tratta di un interprete BASIC esteso, che permette l'utilizzo di variabili reali, intere o alfanumeriche, comprende numerose funzioni sulle stringhe, sui numeri reali, su matrici a più dimensioni etc.

L'implementazione in memoria RAM del programma interpretato dal BASIC Applesoft è diviso in tre zone: la zona P delle istruzioni del programma, situata nella parte bassa della memoria.

La zona V, al di sopra della precedente, che contiene le variabili intere e reali e i puntatori delle stringhe alfanumeriche.

La zona C, delle stringhe di caratteri che è situata nella parte alta della memoria.

Nella zona delle variabili o dei dati numerici, si distinguono: una zona di variabili semplici e, al di sopra, una zona delle matrici di variabili.

Scheda della memoria RAM durante l'esecuzione di un programma scritto in BASIC Applesoft:

Gli indirizzi di inizio e fine di queste zone sono accessibili ai seguenti indirizzi:

Indirizzi Esadecimali	Indirizzi Decimali	Contenuti	Identificazione
$67,$68	103,104	Indirizzo d'inizio della zona del programma. In generale $800	DP
$AF,$B0	175,176	Indirizzo di fine della zona del programma	FP
$69,$6A	105,106	Indirizzo d'inizio della zona dei dati numerici. In generale uguale a FP	LOMEM
$6B,$6C	107,108	Indirizzo d'inizio della zona delle tabelle numeriche	DT
$6D,$6E	109,110	Indirizzo di fine della zona delle variabili numeriche	FN
$6F,$70	111,112	Indirizzo d'inizio della zona delle stringhe di caratteri	DC
$73,$74	115,116	Indirizzo di fine della zona delle stringhe di caratteri	HIMEM

L'APPLE E I SUOI "CUGINI"

Apple II e Euroapple

La scheda principale dell'Apple II è progettata per generare un segnale video NTSC/60Hz incompatibile con le norme europee.

Questa scheda è facilmente modificabile cambiando il quarzo per assicurare lo standard di 50 Hz. e aggiungendo una scheda di interfaccia a colori per la codifica del segnale video nello standard PAL o SECAM.

Apple II e Apple II plus

La scheda principale dell'Apple II contiene una ROM, il Monitor, l'Integer BASIC con il mini-assemblatore e le "utility".

Sostituendo il Monitor con la ROM Autostart e l'Integer BASIC con l'Applesoft in ROM si ottiene l'Apple II Plus.

Apple II e ITT 2020

Il sistema grafico in alta risoluzione dell'Apple II ha una definizione di 280 x 192 punti.

Con l'ITT 2020, questa definizione migliora poichè è possibile visualizzare 360 x 192 punti in colore.

Con questo modello l'elaborazione dei grafici risulta più lenta.

L'Apple II e l'ITT 2020 differiscono solo per la grafica in alta risoluzione, tutte le altre funzioni sono identiche.

Apple II e SILEX (venduto solo in Francia)

Entrambi hanno la stessa scheda principale.

La tastiera del Silex è più completa, poichè dispone delle minuscole, del tastierino numerico separato, dei tasti di funzione (funzione Editing/Correzione, lancio del sistema operativo, accesso al registratore a cassette, alla stampante, etc.).

Il video del Silex è integrato, dotato di fosfori verdi, consente in opzione 80 caratteri per linea.

Il software e le interfacce sono interamente compatibili con tutti i modelli Apple.

CAPITOLO 2

IL BASIC APPLESOFT

Il linguaggio BASIC e le sue estensioni sono state chiamate **Applesoft** dal costruttore dell'Apple II (che propone anche un altro BASIC più semplice chiamato **Integer BASIC**).

Questi due BASIC sono disponibili sia in ROM sia su supporto magnetico (cassetta o dischetto) per il caricamento in RAM.

Proponiamo di analizzare, nella maniera più completa, le istruzioni, le variabili, le funzioni programmabili dell'Applesoft, questo BASIC presenta dei notevoli vantaggi nei confronti dell'Integer BASIC, vantaggi e prestazioni che saranno esaminati nei paragrafi di questo capitolo.

Cominceremo con il rivedere come devono essere concatenate le istruzioni e come devono essere definiti i parametri di un programma scritto in BASIC, saranno poi precisate alcune particolarità di certe istruzioni come IF....THEN.... e le istruzioni di entrata (Input)/uscita (Output) (I/O) in Applesoft.

In seguito saranno rivisti i tipi di variabili e la loro "materializzazione" in memoria.

Un paragrafo farà poi la sintesi di queste definizioni che saranno applicate alla realizzazione di una gestione dati.

LE BASI DELL'APPLESOFT

esempio introduttivo

Si analizzi il seguente programma dato come esempio per differenziare le unità sintattiche del linguaggio e per dettagliare le procedure di esecuzione.

```
10  REM   esempio introduttivo
20  PRINT "come và"
30  INPUT R$
```

```
40   IF R$ = "bene grazie e tu?"          THEN N = 1: GOTO 80
50   IF R$ = "abbastanza bene e tu?"           THEN N = 2: GOTO 80
60   IF R$ = "male"   THEN N = 3: GOTO 80
70  N = 0
80  L$  =  "tutto và per il meglio"           _IO"
90  M$ = ".....":N$ = ".....":P$  = "....."
100   IF N = 1 THEN  PRINT L$ : GOTO 140
110   IF N = 2 THEN  PRINT M$: GOTO 140
120   IF N = 3 THEN  PRINT N$: GOTO 140
130   IF N = 0 THEN  PRINT P$: GOTO 140
140   END
```

Si distinguono tre categorie di entità:

- in MAIUSCOLO sono le parole riservate del linguaggio Applesoft che corrispondono a degli ordini, dunque a operazioni che devono essere eseguite:

PRINT	visualizzazione sullo schermo
INPUT	attesa di dati dalla tastiera
GOTO	salto incondizionato, come dire rottura della sequenza naturale delle istruzioni
IF...THEN	test di una espressione logica
END	arresto dell'esecuzione di un programma
REM	identificazione di una linea di commento che non viene eseguita.

Nella stessa categoria, si notano dei *caratteri riservati* che provocano specifiche operazioni:

	concatenamento di più istruzioni sulla medesima linea di programma
$	specifica un tipo di variabile (alfanumerica)
=	assegnazione di un valore ad una variabile o segno di comparazione
" "	identificazione di una costante alfanumerica.

- In caratteri MINUSCOLI sono definite le costanti alfanumeriche o numeri per la rappresentazione di costanti numeriche.
  Questa è la categoria dei valori costanti introdotti nel programma dall'operatore,

se questi valori vengono modificati non cambia la struttura del programma, ma cambiano i risultati che si ottengono dopo l'esecuzione dello stesso.

- Ancora in MAIUSCOLO corsivo si riconoscono le variabili di programma identificate da un nome e differenziate nei vari tipi:

R$ è la variabile alfanumerica che consentirà di rispondere alla domanda della linea 20.

P$,T$ sono le variabili alfanumeriche utilizzate per memorizzare le differenti risposte possibili da parte del calcolatore.

N è la variabile numerica che servirà ad indicare l'umore dell'operatore.

Questa analisi delle istruzioni di un programma (detta analisi lessicografica) è realizzata dall'interprete Applesoft (più precisamente da uno dei suoi numerosi sottoprogrammi: lo "*lo scanner*") prima che sia realmente posta in atto l'esecuzione del programma stesso.

Analizziamo ora passo-passo l'esecuzione delle istruzioni del programma.

Tutti sanno che è obbligatorio, in BASIC, numerare ciascuna linea di programma.

La numerazione definisce, in modo categorico, la sequenza delle operazioni da eseguire (in ordine crescente rispetto ai numeri di linea).

L'istruzione REM della linea 10 non prende parte all'esecuzione, essa serve solo per l'impostazione di commenti quando si scrive un programma.

Dunque, dopo la visualizzazione del messaggio "Come va ?", è attesa una risposta, che dovrà essere impostata sulla tastiera. Questo è dovuto all'istruzione INPUT *R$*.

Dopo che l'operatore avrà risposto da tastiera e avrà premuto il tasto RETURN, la variabile *R$* conterrà la risposta sotto forma di una serie di caratteri alfanumerici.

Data questa serie di caratteri, le istruzioni delle linee 40, 50, 60, comparano *R$* a dei messaggi preregistrati.

Quando è ottenuta una stringa uguale, le altre comparazioni sono evidentemente saltate, sarà effettuata una selezione delle risposte date dal calcolatore con le

istruzioni delle linee 100, 110, 120, 130, e sarà dato un solo ordine di visualizzazione.

La variabile numerica *N* è l'indicatore: questa riceve un valore (3, 2, 1 oppure 0) nel corso delle comparazioni e tale valore è poi testato nel corso della selezione.

Per ottenere la visualizzazione sullo schermo del contenuto della variabile *L$* è sufficiente una istruzione PRINT *L$*.

Vediamo in pratica come agisce il programma:

```
]RUN
COME VA  ?
?bene grazie e tu?
TUTTO VA  PER IL MEGLIO

]
```

I NOMI DELLE VARIABILI

Il nome di una variabile è soggetta alle seguenti regole:

- Il nome deve cominciare con un carattere alfabetico e può essere seguito da qualsiasi altro carattere alfanumerico (lettere o cifre).

- Il numero massimo dei caratteri è 238. Ma l'interprete Applesoft utilizza soltanto i primi due caratteri per distinguere una variabile da un'altra.

- Tutti i caratteri alfanumerici che seguono i primi due sono ignorati, a meno che non contengano una delle parole riservate (vedere l'appendice II, elenco delle istruzioni). Le parole riservate sono, in Applesoft, delle istruzioni e il loro impiego, in un nome di variabile, rischia di modificare totalmente la logica delle istruzioni del programma.

- La specifica del tipo di variabile è soggetta alle seguenti regole:

  . Nome della variabile per le variabili reali

  . Nome seguito dal carattere percento (%) per le variabili intere

  . Nome seguito dal carattere dollaro ($) per le variabili alfanumeriche o stringhe di caratteri.

A, A$, A% sono tre variabili distinte

Esempio:

ME$ = "SI" } Queste due variabili non saranno differenziate
MESE$ = "APRILE" } durante l'esecuzione del programma.

```
]10 PLATONE$ = "IL BANCHETTO"

]LIST 10

10 PLA TO NE$ = "IL BANCHETTO"

]RUN

?SYNTAX ERROR IN 10
```

LE ISTRUZIONI DI ENTRATA/USCITA (I/O)

Lo scambio di informazioni tra la memoria RAM e i dispositivi di I/O sono realizzate per classi di istruzioni:

- La *prima classe* contiene le istruzioni classiche di tutti i linguaggi BASIC:

INPUT
GET
PRINT

La selezione delle periferiche, per queste istruzioni, può essere sia implicita (entrata da tastiera e uscita su video), sia esplicita con comandi del tipo:
IN # s
PR # s

dove "s" è il numero del connettore sul quale è collegata la scheda di controllo della periferica selezionata.

- La *seconda classe* contiene degli ordini particolari per ciascun dispositivo connesso all'Apple.

. L'altoparlante attivato con:
X = PEEK (-16336)

. Un segnale è registrato su cassetta con:
X = PEEK (-16352)

- La posizione delle paddle è rilevata con:
  X = PDL (0)
  Y = PDL (1)

- Lo stato dei pulsanti delle paddle è testato con:
  X = PEEK (-16287)
  Y = PEEK (-16286)

- Il registro di uscita della tastiera è visibile, ed il suo contenuto controllabile o trasferibile in memoria con:
  X = PEEK (-16384)
  POKE -16368,0 per leggere un'altro carattere.

SINTASSI DELL'ISTRUZIONE INPUT

INPUT ["messaggio"]; nome della variabile [,nome della variabile] (le parentesi quadre racchiudono gli elementi opzionali).

- Se dev'essere visualizzato un messaggio, allora *il punto e virgola è obbligatorio* prima dell'impostazione del nome della variabile.

- Se più dati devono essere impostati in serie, allora *la virgola è obbligatoria* per separare i nomi delle variabili.

Esecuzione di un'istruzione INPUT

Il programma si arresta a questa istruzione.

Lo schermo visualizza all'inizio della linea:

sia il messaggio, se esiste nell'istruzione,

sia un punto di domanda.

La tastiera è attiva, il dato impostato viene visualizzato sullo schermo e memorizzato solo dopo l'impostazione di un RETURN.
Dopo l'acquisizione del dato, il programma prosegue.

Effetti specifici di alcuni caratteri

Carattere impostato	Effetto:
RETURN o CTRL-M	fine assegnazione e proseguimento del programma
“,“	fine assegnazione e concatenamento con la lettura seguente
CTRL-X	annullamento del valore impostato (solo prima del RETURN)
CTRL-C	arresto forzato del programma

Messaggi di errore nell'esecuzione di INPUT

?REENTER - Per un errore sul tipo di dato da assegnare.

? - Un dato numerico è sempre accettato anche se associato ad una variabile alfanumerica.

- Mai un dato alfanumerico entrerà in memoria associato al nome di una variabile numerica.

- Questo messaggio domanda dunque di reimpostare un dato numerico.

?EXTRA IGNORED - i dati (separati da virgole) sono in numero superiore a quello delle variabili che li devono ricevere, questo messaggio indica che i dati in eccesso sono ignorati, e il programma prosegue.

- Se è impostato il carattere “:“ questo impedisce tutti gli assegnamenti successivi. Solo un RETURN riprende il controllo.

I due modi di assegnare i dati multipli

L'istruzione INPUT A,B,C è equivalente alle istruzioni

```
INPUT A
INPUT B
INPUT C
```

Per l'esecuzione dell'istruzione INPUT A,B,C sono applicabili le seguenti formule:

Prima formula ?10 'RETURN'
??5 'RETURN'
??0 'RETURN'

dove i dati, a partire dalla seconda variabile, sono attesi da ??.

Seconda formula ?10,5,0 'RETURN'

la virgola indica la fine di un dato e il concatenamento con il seguente.

Esempio

```
10    INPUT A,B$
20    GOTO 10
RUN
```

Azioni	Effetti
?Ø 'Return'	A=Ø
?? 'Return'	BØ = "" (stringa vuota)
?1Ø, OUI 'Return'	A=1Ø :BØ="OUI"
?-1," NON" 'Return'	A=-1 :BØ=" NON"
?Ø, NON 'Return'	A=Ø :BØ="NON"
?A'Return'	
?REENTER	errore sul tipo di variabile
?A'Ctrl X'	correzione del carattere battuto
6 'Return'	A=6
??1.Ø7 'Return'	BØ ="1.Ø7"
?Ø.Ø1,Y,Z	
?EXTRA-IGNORED	A=Ø.Ø1 : BØ = "Y"
?'Ctrl C'	
BREAK IN 1Ø	arresto forzato

Gli spazi che precedono dati numerici sono ignorati. Gli spazi che precedono dati alfanumerici non inseriti tra virgolette sono anch'essi ignorati.

L'ISTRUZIONE GET

L'istruzione GET R$ arresta il programma fino a quando non viene impostato un carattere da tastiera.

È atteso solo un carattere, e qualunque sia il suo valore, il programma proseguirà.

Inoltre il carattere impostato non sarà visualizzato sullo schermo.

Si potranno impostare dati che includono anche caratteri speciali (, : RETURN) che non sono accettati dall'istruzione INPUT.

Esempio:

```
10  GET C$
15  IF C$ = "CTRL C" THEN  END
20 A$ = A$ + C$
25  GOTO 10

]RUN

]PRINT A$
QWERTY
```

L'ISTRUZIONE PRINT

Grazie a questa istruzione, tutti i risultati (o tutte le variabili) sono estratti dalla memoria per essere visualizzati sullo schermo.

Per visualizzare più valori sulla stessa linea, è necessario utilizzare, come separatore di variabili il punto e virgola (;) che divide le informazioni, oppure la virgola che visualizza le informazioni in zone prestabilite:

- la prima zona contiene 16 caratteri,
- la seconda zona contiene 16 caratteri,
- la terza zona contiene 8 caratteri.

Se la larghezza di una zona non è sufficiente il successivo valore sarà visualizzato nella zona che segue interamente disponibile.

Esempio

```
10  PRINT 2;3;4
20  PRINT 2,3,4
30  PRINT "A";"B";"C"
40  PRINT "A................","B"
```

```
50  INPUT C$
60  PRINT C$;
70  GOTO 50

]RUN
234
2                   3                   4
ABC
A.................                      B
?X
X?Y
Y?Z
Z?CTRL C
```

Il punto e virgola o la virgola di un'istruzione PRINT impedisce il ritorno automatico alla linea seguente.

Esistono delle funzioni di tabulazione e regolazione della dimensione dello schermo che permettono una migliore visualizzazione (vedere pag. 68, 69)

L'ISTRUZIONE IF...THEN

In Applesoft l'istruzione IF...THEN:

- contiene, tra IF e THEN, l'espressione logica il cui valore VERO o FALSO condurrà all'esecuzione condizionale di alcune istruzioni.

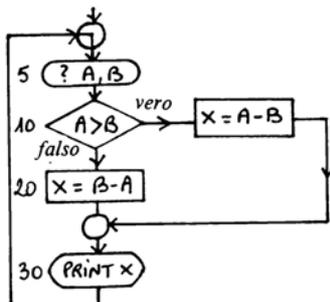
- Seguita dopo THEN, da una o più istruzioni separate da : (due punti separano due istruzioni su una stessa linea) o da un semplice numero di linea. Queste istruzioni saranno eseguite solo se l'espressione logica è VERA.

- Alla linea numerata, successiva all'istruzione IF...THEN, iniziano le istruzioni che saranno eseguite se l'espressione logica è FALSA.

Se l'espressione logica è falsa, tutte le istruzioni che seguono THEN, vengono ignorate fino alla successiva istruzione numerata.

```
5  INPUT A,B
10  IF A > B THEN X = A - B: GOTO
    30
20 X = B - A
30  PRINT X: GOTO 5

]RUN
```

```
?3,1
2
?1,3
2
?CTRL C
```

L'ISTRUZIONE FOR...NEXT

In BASIC Applesoft, l'istruzione FOR comprende:

- Il nome della variabile che sarà modificato automaticamente ad ogni iterazione, i valori consentiti e l'incremento o decremento da realizzare sulla variabile ad ogni iterazione.

FOR I = -100 TO 100 STEP 2

permette la variazione di I di 2 in 2 dal valore -100 al valore +100.

- Il ciclo del loop termina con NEXT, *nome della variabile* dichiarata in FOR.

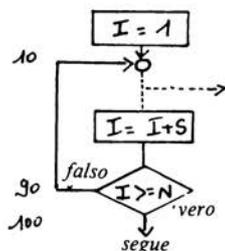
```
10  FOR I = 100 TO 100 STEP 2
20  PRINT I + 1,2 * I - 1
30  NEXT I
```

Se il valore limite inferiore è superiore al valore limite superiore con uno STEP positivo, le istruzioni tra FOR e NEXT saranno eseguite una sola volta.

Esiste un solo punto di entrata autorizzato in un ciclo FOR...NEXT; si trova all'inizio, cioè all'istruzione FOR.

Quando la variabile ha raggiunto il suo valore limite superiore, il programma prosegue con l'istruzione che segue il NEXT.

Si può uscire dal ciclo prima della sua normale conclusione per mezzo di un'istruzione condizionale.

```
10   FOR I = 1 TO N STEP S
11   REM  ISTRUZIONI
90   NEXT I
100   REM
```

LE ISTRUZIONI READ...DATA

In un programma scritto in Applesoft alcuni dati possono essere utilizzati permanentemente senza che siano stati presi da una sola variabile.

Questi dati si chiamano **DATA** e costituiscono un insieme sequenziale nella memoria RAM.

Sintassi della lista dei dati

In seguito alla parola riservata DATA, si scrivono i dati (costanti alfanumerici o numerici) separati dalle virgole.

I dati seguono le stesse regole dell'istruzione INPUT.

DATA SI,"NO",12,0.01

Esecuzione dell'istruzione READ

Quando l'istruzione **READ** è eseguita per la prima volta, la sua prima variabile riceverà il primo valore della lista dei DATA, la seconda variabile riceverà il secondo valore della lista DATA.

Quando l'istruzione READ termina, mantiene il puntatore dei dati dopo il primo valore letto, l'istruzione che segue READ utilizzerà dunque i restanti dati della lista.

L'istruzione **RESTORE** forza il puntatore al primo elemento della lista.

Messaggio d'errore

?OUT OF DATA ERROR segnala che mancano dei dati nella lista che dev'essere letta dall'istruzione READ.

Esempio

```
5  INPUT J
10  FOR I = 1 TO J
15  READ J$
20  DATA   LUNEDI',MARTEDI',MERCOLEDI',GIOVEDI',
           VENERDI',SABATO,DOMENICA
30  NEXT I
40  PRINT J$
50  RESTORE
60  GOTO 5

]RUN
?3
MERCOLEDI'
?7
DOMENICA
?8

?OUT OF DATA ERROR IN 15
]NEW
```

```
10  READ N
15  FOR I = 1 TO N / 2
20  READ X,Y
30  PRINT X;
40  NEXT I
50  DATA  10,0,0,1,0,1,0,1,1,0,1

]RUN
01110
]
```

Per poter leggere una sola assegnazione ogni due, è obbligatorio leggere l'assegnazione presente e quella successiva per spostare il puntatore di una posizione.

Il primo dato di questo esempio è il numero delle assegnazioni contenute nella lista DATA.

GLI OPERATORI ARITMETICI E LOGICI

Ecco i ordine di esecuzione delle operazioni dalla più importante alla meno importante:

( )	Parentesi
+ — NOT	Operazioni su un solo operando
∧	Elevazione a potenza
* /	Moltiplicazione, divisione
+ —	Addizione, sottrazione
> <>= <= => =< <> <>=	Comparazioni
AND	AND logico
OR	OR logico

Gli operatori aventi la medesima priorità saranno eseguiti da sinistra a destra nell'espressione da calcolare.

Esempio

```
10 I = 0
20  IF  NOT I THEN 200
30  REM  SERIE A D'ISTRUZIONI
40 I = 0
50  GOTO 20
200  REM  SERIE B D'ISTRUZIONI
210 I = 1
220  GOTO 20
```

Il programma eseguirà alternativamente prima le istruzioni B poi le istruzioni A.

I SOTTOPROGRAMMI

L'istruzione GOSUB 100 permette l'esecuzione di un sottoprogramma che inizia alla linea di istruzione 100 e termina con un'istruzione RETURN. Dopo l'esecuzione del sottoprogramma, il programma principale prosegue con l'istruzione che segue GOSUB 100.

In effetti, con l'esecuzione del GOSUB, l'indirizzo relativo all'istruzione successiva è posizionato in cima allo "STACK" degli indirizzi di ritorno.
Alla fine del sottoprogramma, l'indirizzo di ritorno viene utilizzato e cancellato dallo "STACK".

Questo spostamento può anche essere forzato per mezzo dell'istruzione POP.

Il numero massimo degli indirizzi di ritorno è fissato in 25. Ciò significa che sono **programmabili 25 livelli di chiamata**. Oltre questo valore apparirà il messaggio ?OUT OF MEMORY ERROR.

Per programmare una funzione ricorrente (che chiama se stessa) sarà necessario prevedere, in testa al sottoprogramma, la protezione delle variabili locali su una "catasta" e alla fine del sottoprogramma il ripristino delle variabili situate in cima alla catasta stessa.

L'istruzione ON I GOSUB 100,200,300 permette di saltare ad uno dei sottoprogrammi indicati, 100 se I = 1, 200 se I = 2, 300 se I = 3.

I NUMERI REALI

I lettori che non hanno familiarità con le numerazioni binarie ed esadecimali possono consultare, prima di proseguire la lettura, l'appendice "Richiami sulle basi della numerazione binaria ed esadecimale"e" a pag. 117.

Rappresentazione interna

Come ogni unità aritmetica del calcolatore, quella dell'Apple opera su registri di calcolo di lunghezza fissa e limitata.

La lunghezza dei registri è di 8 bit o di un byte. Vi è la possibilità di aumentare questa lunghezza, la precisione e l'estensione dei risultati di calcolo facendo intervenire dei sottoprogrammi di aritmetica estesa.

I numeri interi, per esempio, sono registrati in memoria con 2 byte, i numeri reali sono registrati con 5 byte.

Studiamo il loro principio di rappresentazione.

Rappresentazione dei numeri interi

Gli interi positivi sono rappresentati da una semplice numerazione binaria su 15 bit, e quindi la loro estensione va da 0 a $2^{15} - 1 = 32767$.

Il bit più a sinistra, quindi il sedicesimo, è posto a 0.

Gli interi negativi sono rappresentati dal loro complemento a $2^{14} = 65536$.

$$-1 \text{ diviene } (65535)_{10} = (FFFF)_{16}$$
$$-2 \text{ diviene } (65534)_{10} = (FFFE)_{16}$$
$$-32168 \text{ diviene } (32768)_{10} = (8000)_{16}$$

Perchè si utilizza questa rappresentazione? Per facilitare le operazioni aritmetiche, la sottrazione si esegue con l'addizione del complemento e l'eliminazione del bit più significativo (peso 2^{16}).

Il bit più a sinistra è sempre a 1 per gli interi negativi.

Per ogni operazione aritmetica con numeri interi si devono prevedere dei risultati inclusi nell'intervallo

$$\{-32768, +32767\}$$

con un errore massimo di 0,5 nel risultato di una divisione intera.

Rappresentazione interna dei numeri reali

Si sa che la notazione detta "a virgola flottante" diminuisce considerevolmente il numero delle cifre per la rappresentazione di un numero frazionario o reale.

La Virgola flottante significa che sono utilizzate solo le cifre rappresentative, c'è la mantissa M, ma a queste cifre si aggiunge un esponente E che caratterrizza la posizione della virgola del numero reale.

$$0{,}0001 = 1 \times 10^{-4} \rightarrow (M=1,\ E=-4)$$
$$100000 = 1 \times 10^{5} \rightarrow (M=1,\ E=-5)$$

In Applesoft è possibile registrare in memoria dei numeri reali aventi fino a 9 cifre

significative che possono essere, in valore assoluto:

PICCOLE FINO A 10^{-38}
GRANDI FINO A 10^{+38}

Per ottenere una certa precisione e un certo controllo di variazione, i numeri reali sono codificati in memoria con 5 byte ripartiti in un byte per l'esponente e 4 byte per la mantissa.

E è il valore decimale dell'esponente,
e M il valore decimale della mantissa
e N il valore decimale del numero rappresentato.

$$-127 < E <= 127$$
$$2^{-32} < M < 0,5$$
$$\boxed{N = (M+0,5)\ 2^{E-128}}$$

questa è la formula che permetterà di determinare il valore assoluto di N rappresentato in memoria da E e M.

Esempi:

Nel byte dell'esponente, si trova il valore esadecimale $(E)_{16}$ e nei 4 byte della mantissa, supponiamo $(M)_{16}$, il loro valore esadedecimale.

Bisogna eseguire la transposizione di $(E)_{16} \rightarrow E_{10}$ e $M_{16} \rightarrow M_{10}$ prima di calcolare N_{10}

```
(E) 16          (M) 16
|8 1|   |Ø Ø | Ø Ø | Ø Ø | Ø Ø|
```

E_{10} = 8 x 16+1 = 129
E-128 = +1
2^{E-128} = 2
M_{10} = Ø
N = (Ø.5)*2 = +1

(E) 16 (M) 16

| 8 7 | | 4 8 | Ø Ø | Ø Ø | Ø Ø |

$E_{10} = 8 \times 16+7 = 135$
$E-128 = 7$
$2^{E-128} = 128$
$M_{1Ø} = 4*16^{-1} + 8*16^{-2} = 0,28125$
$N = (0,78125)*128 = 1ØØ$

(E) 16 (M) 16

| 8 2 | | C Ø | Ø Ø | Ø Ø | Ø Ø |

$E_{10} = 8 \times 16+2 = 130$
$E-128 = 2$
$2^{E-128} = 4$

(M) 16 ha il bit piu' a sinistra a 1.
Cio' indica un numero negativo.

$M_{10} = (C-8)*16^{-1} = 4x16^{-1} = 0.25$
$N_{10} = -(0,25+0,5)\,2^{2} = -3$

TABELLA DEI VALORI LIMITE ESATTI
(9 cifre significative)

	VALORE DECIMALE		CODIFICA INTERNA	
	N 10	N Virgola flottante	E_{16}	M_{16}
Numero positivo piu' piccolo	$1,7Ø141183.10^{38}$	$\sim 2^{128}$	FF	7FFFFFFF
N > Ø	$8,5Ø7Ø5917.10^{37}$	$0,5.2^{127}$	FF	00000000
Numero positivo piu' piccolo	$2,93873588.10^{-39}$	$0,5.2^{-127}$	01	00000000
N = Zéro			00	00000000
Numero negativo piu' piccolo	$-2,93873588.10^{-39}$	$-0,5.2^{-127}$	01	80000000
N < Ø	$-8,5075917.10^{37}$	$-0,5.2^{+127}$	FF	80000000
Numero negativo piu' piccolo	$-1,7Ø141183.10^{38}$	$\sim 2^{+128}$	FF	FFFFFFFF

PRECISIONE DI ARROTONDAMENTO

Come dalla tabella sopra riportata, tutte le operazioni in virgola flottante, dovranno fornire un risultato compreso tra -1.7 10^{38} e +1.7 10^{38}, in caso contrario sarà visualizzato il messaggio:
OVERFLOW ERROR.

Un numero il cui valore assoluto è più piccolo di 2,9 10^{39} sarà automaticamente equiparato a 0.

Poichè la codifica della mantissa è su 32 bit, i calcoli sono precisi fino a 2^{-32}, questo implica che la registrazione in memoria ha una precisione più elevata rispetto alle nove cifre significative visualizzate (2^{-32}=0.000 000 000 2).

SINTASSI E VISUALIZZAZIONE DEI NUMERI REALI

I valori costanti dati alle variabili del tipo reale possono essere scritti normalmente o in notazione flottante.

Questi valori utilizzano la lettera E quale simbolo del fattore 10 che sarà elevato a una potenza il cui valore segue la lettera E dell'esponente.

Il valore scritto alla sinistra di E moltiplicato per la potenza di 10, fornirà il valore risultante del numero.

Esempi

0,0001	si scrive		0.0001
		oppure	1 E-4
1.000.010,05	si scrive		1.000 010.05
		oppure	1.000 010 05 E6
13,9.10^{-9}	si scrive		13.9 E-9
		oppure	.139 E-11

Per la visualizzazione dei numeri reali, l'Applesoft mostra solitamente 9 cifre significative, i caratteri e i valori di esponente non sono compresi in questo numero.

A sinistra del punto decimale, tutti gli zeri che precedono cifre significative più a sinistra, non sono visualizzati.

A destra del punto decimale, tutti gli zeri che seguono cifre significative più a destra non sono visualizzate.

Se non vi sono cifre significative a destra del punto decimale, il punto stesso non è visualizzato.

Se 0,01 <= /N/ < 999 999 999,2 ; N sarà visualizzato normalmente (virgola fissa) senza esponente.

Esempi

Limitazione del numero dei decimali visualizzati di un numero X come 1 <= X < 999 999 999

Supponiamo D il numero dei decimali impostati.

Moltiplicando X per 10^D, poi arrotondando il risultato al numero intero più prossimo, si ottiene il numero intero cercato ma "10 elevato alla D volte" più grande.

Per esempio:

X = 1,3779 e D = 2
1,3779 x 10^2 = 137,79
arrotondando 137,79 = 138

Si divide quindi per 10^D
138/100 = 1,38
la formula è:

X=INT (X*10 ∧ D + 0.5)/10 ∧ D

∧ è il simbolo di elevazione a potenza.
INT è la funzione che consente di ottenere un valore intero.

Sia l'arrotondamento che la visualizzazione, non influiscono sul valore reale registrato in memoria:

```
10  PRINT "1/3*3 = ";1 / 3 * 3
20  PRINT "1/3 = ";1 / 3
30  PRINT ".333 333 333*3 = ";.333333333 * 3
40  PRINT ".333 333 333 3*3 = ";.3333333333 * 3

]RUN
1/3*3 = 1
1/3 = .333333333
.333 333 333*3 = .999999999
.333 333 333 3*3 = 1
```

Generazione dei più grandi numeri decimali possibili

```
100 X = 1.6999999E38
110  PRINT X
120 X = X + 1E33
130  GOTO 110
```

Questo programma visualizza il numero più grande e si ferma con il messaggio OVERFLOW ERROR.

Generazione dei più piccoli numeri decimali possibili

```
200 X = 1E - 30
215  PRINT X
220 X = X / 2
230  IF X <  > 0 THEN 215
240  END
```

dei quali $X <= 2.93873588.10^{-39}$ è equivalente a zero.

Come leggere in memoria i codici dei numeri reali.

```
]NEW
]10 A=1        registrazione del valore 1
]RUN           esecuzione
]CALL-151      passaggio al Monitor
*800.817

0800-  00 09 08 0A 00 41 D0 31
0808-  00 00 00 01 41 00 81 00
0810-  00 00 00 20 50 41 47 2E
```

L'esame delle locazioni di memoria che contengono il codice dell'istruzione della linea 10, il codice della variabile A e del suo contenuto nella zona seguente riservata ai dati, si esegue in questo modo:

0800	Indirizzo di inizio della zona programma,
0801 e 0802	contengono il byte meno significativo e il byte più significativo dell'indirizzo di inizio del codice relativo alla istruzione successiva (809),
0803 e 0804	contengono il numero di istruzioni cominciando dal byte meno significativo $((0A_{16} = (10)_{10})$.

0805	Contiene il codice ASCII del carattere “A“
0806	Contiene il codice ASCII del carattere “=“
0807	Contiene il codice ASCII del carattere “1“
080C e 080D	Contengono il codice ASCII del nome “A“
080E	Contiene l’esponente
da 080F a 812	Contengono la mantissa.

Esercizi

2.1 Scrivere un programma con una sola istruzione 10 A%=4096 e ritrovare la codifica esadecimale corrispondente a 4096.

2.2 Cambiare il contenuto della memoria riservata al valore dell’intero con il valore esadecimale F000. Chiedere poi, da Applesoft: PRINT A% per trovare il valore intero corrispondente a F000.

2.3 Convertire 2 elevato alla 15ma nella relativa codifica interna (esponente - mantissa). Convertire 00 FF FF FF FF in notazione a virgola flottante.

2.4 Avete notato che: (12345678 + 1) x 9 = 111 111 111
(12345678 + 1) x 18 = 222 222 222 etc.

Scrivere un programma di generazione automatica di queste operazioni.

LE STRINGHE DI CARATTERI

Definizioni - caratteristiche - concatenazione - occupazione di memoria

Le informazioni non numeriche sono registrate e codificate in memoria, carattere per carattere; ciascun carattere è una combinazione unica di sette bit determinata per mezzo del codice ASCII (American Standard Code for Information Interchange - Appendice III).

Per manipolare sequenze di caratteri si utilizzano particolari tipi di variabili chiamate variabili alfanumeriche che hanno per valore delle stringhe di caratteri.

Per definizione, una stringa di caratteri è costituita da una serie di caratteri

elementari, e la lunghezza di una stringa di caratteri è uguale al numero dei caratteri che la compongono.

La registrazione di un dato alfanumerico si ottiene assegnando ad una variabile alfanumerica una stringa di caratteri "racchiusa" fra le virgolette, per esempio:

A$ = "APPLE"

La variabile di tipo alfanumerico si specifica facendo seguire al nome della variabile il carattere dollaro ($).

La lunghezza massima autorizzata per una variabile alfanumerica è di 255 caratteri.

La lunghezza minima di una stringa di caratteri è zero, essa non contiene alcun carattere e si chiama stringa vuota.
La valutazione della lunghezza di una variabile alfanumerica è possibile con la funzione: L = LEN (A$).

Per esempio:

```
] PRINT LEN ("APPLE")
  5
] PRINT LEN (" ") virgolette adiacenti
  0
```

Concatenazione di stringhe di caratteri

Un'operazione elementare di elaborazione per le stringhe di caratteri consiste nell'unire due stringhe (o concatenare una stringa ad un'altra). Questa operazione si effettua con l'operatore + (l'operazione non è commutativa).

Per esempio:

```
10 E$ = " ": REM   SPAZIO
15  INPUT A$
20 A$ = E$ + A$
25 A$ = A$ + "?"
30  PRINT A$

]RUN
?APPLE
 APPLE? (uno spazio in testa)
```

La nuova stringa costituita non deve avere una lunghezza superiore a 255 caratteri.

Un artificio consente di superare questo limite se si desidera visualizzare una stringa che composta risulti troppo grande, nell'istruzione PRINT si eliminano gli operatori +, quindi PRINT ABC$.

Occupazione in memoria di una stringa di caratteri

Una variabile alfanumerica è caratterizzata dal suo nome (2), dalla lunghezza (1) della stringa di caratteri che gli è stata assegnata e dall'indirizzo (2) partendo da dove questa stringa è posizionata nella memoria, indirizzo seguito da due byte nulli.

Dunque per ciascuna variabile alfanumerica, lo spazio occupato in memoria è di 7 byte di identificazione oltre al numero dei caratteri occupati che compongono la stringa stessa.

Ma se a questa variabile alfanumerica si desidera assegnare un altro valore, allora questo secondo valore è posto nella prima zona disponibile, dunque ad un nuovo indirizzo, i parametri di indirizzo e la lunghezza sono modificati senza che il precedente valore alfanumerico sia cancellato. La vecchia stringa è così abbandonata e occupa, inutilmente, dello spazio in memoria.

Analisi dell'implementazione di un programma e dei suoi dati alfanumerici

```
10 E$ = " "
15  INPUT A$
20 A$ = E$ + A$
25 A$ = A$ + "?"
30  PRINT A$

]RUN
?APPLE
 APPLE?  (un

]CALL-151   passaggio al monitor

*69.6A      puntatore d'inizio della zona dati

0069- 3A 08
*800.839    zona delle istruzioni di programma

0800- 00 0C 08 0A 00 45 24 D0
0808- 22 20 22 00 14 08 0F 00
0810- 84 41 24 00 21 08 14 00
0818- 41 24 D0 45 24 C8 41 24
0820- 00 2F 08 19 00 41 24 D0
```

```
0828- 41 24 C8 22 3F 22 00 37
0830- 08 1E 00 BA 41 24 00 00
0838- 00 01
```

La linea di istruzione 10 è codificata nei byte di indirizzo da 0801 a 080B:

- 801 - 802 contengono il valore 08 0C, indirizzo di inizio dell'istruzione successiva;

- 803 - 804 contengono il numero dell'istruzione 0A

- 805 - 806 contengono il nome della variabile E$; '45 24'

- 807 contiene il codice ASCII del segno =; 'D0'

- 808 - 80A contengono la serie " " codificata; '22 20 22'.

```
*83A.845        zona riservata ai dati

083A- 45 80 01 09 08 00
0840- 00 41 80 07 EE 95
```

La prima variabile del programma è caratterizzata dal suo nome, dal tipo, dalla lunghezza e dall'indirizzo dei suoi dati:

- 083A e 083B contengono i codici ASCII dei caratteri del suo nome, il secondo è negativo per specificare il tipo alfanumerico.

- 083C contiene la lunghezza, in questo caso 1 carattere.

- 083D e 083E contengono l'indirizzo del valore che per questa variabile è 0809. Si verifica che 0809 contiene 20, cioè il codice ASCII dello spazio.

Esaminiamo ora il caso della variabile A$.
Nella zona dati si trovano i parametri che la riguardano:

- 841 - 842 contengono i codici dei caratteri del suo nome, in questo caso '41 80'.

- 843 contiene la lunghezza, in questo caso 7.

- 844 - 845 contengono l'indirizzo del suo valore, in questo caso BF EE.

```
*95EE.95FF    zona riservata alle stringhe di caratteri

95EE- 20 41
95F0- 50 50 4C 45 3F 20 41 50
95F8- 50 4C 45 41 50 50 4C 45
```

La stringa 'spazio'APPLE? è codificata partendo dall'indirizzo 95EE su sette caratteri come previsto.

E' da notare che le altre versioni di A$ prima della sua versione finale, sono rimaste nella parte alta della zona stringhe di caratteri:

- da 95FB a 95FF: 'APPLE' impostato con INPUT A$

- da 95F5 a 95FC: 'spaAPPLE' formato da A$=E$+A$.

Cancellazione delle stringhe di caratteri abbandonate

L'istruzione PRINT FRE(0) consente, non solo di liberare delle zone occupate da stringhe abbandonate, ma informa sullo spazio di memoria ancora disponibile.

Dopo l'esecuzione del programma analizzato precedentemente, impostare

```
]PRINT FRE(0)
-29263
```

Il valore è espresso per complemento a 65536 e vale esattamente 65536+FRE(0)=47025 byte liberi.

```
] CALL - 151

*95EE.95FF

95EE- 20 41
95F0- 50 50 4C 45 3F 20 41 50
95F8- 50 20 41 50 50 4C 45 3F
*840.848

0840- 00 41 80 07 F9 95 00 00
0848- 08
```

L'ultimo valore di A$ si ritrova nella parte alta della memoria e l'indirizzo di questo valore è modificato nella zona delle variabili.

*6F.70 puntatore del più alto indirizzo riservato alla zona stringhe di caratteri.

006F - F9

0070 - 95

Tutti i nuovi valori delle stringhe di caratteri saranno ora registrati in memoria partendo dall'indirizzo 95F9.

IMPORTANTE: l'istruzione FRE(0), utilizzata all'interno di un programma, ottimizza lo spazio occupato in memoria ma ciò richiede un tempo di esecuzione piuttosto lungo quando le stringhe da eliminare sono numerose.

Estrazione di sotto-stringhe

Si distinguono quattro funzioni riservate alle stringhe di caratteri che ricavano una sotto-stringa da una stringa di caratteri:

- *LEFT*$ (A$,I) estrae gli 'I' caratteri più a sinistra di A$.

- *RIGHT*$ (A$,J) estrae gli 'J' caratteri più a destra di A$.

- *MID*$ (A$,K,L) estrae gli 'L' caratteri cominciando dal K*iesimo* carattere.

- *MID*$ (A$,M) estrae una sotto-stringa da A$ cominciando dal M*iesimo* carattere e terminando alla fine di A$.

La prima posizione di una stringa corrisponde a K = 1. L'ultima posizione è data da: K = LEN (A$)

Esempio:

Stringa invertita

```
10  INPUT A$
20  FOR I =  LEN (A$) TO 1 STEP  - 1
30  PRINT  MID$ (A$,I,1)
40  NEXT I
```

Stringa doppia

```
10  INPUT A$
20 A$ = A$ + A$
30  PRINT A$
```

Caratteri doppi

```
10  INPUT A$
20 L =  LEN (A$)
30  FOR I = 1 TO L
40 B$ = B$ +  MID$ (A$,I,1) +  MID$ (A$,I,1)
50  NEXT I
60  PRINT B$
```

Analisi di una frase parola per parola

```
10  INPUT "FRASE ?";F$
20 K1 = 1
30  FOR I = K1 TO  LEN (F$)
40  IF  MID$ (F$,I,1) = " " THEN K2 = I: GOTO 100
50  NEXT I
60  PRINT  MID$ (F$,K1, LEN.(F$) - K1 + 1)
70  END
100  PRINT  MID$ (F$,K1,K2 - K1 + 1)
110 K1 = K2 + 1
120  GOTO 30
```

Soppressione del carattere di una stringa

Il carattere stabilito dev'essere eliminato ogni volta che compare, salvo che si trovi alla fine della stringa:

```
10  INPUT "CARATTERE DA SOPPRIMERE ?";C$
20  INPUT "STRINGA ?";A$
30 B$ = "": REM  B$ E' VUOTO
40  FOR I = 1 TO  LEN (A$)
50 X$ =  MID$ (A$,I,1)
60  IF X$ = C$ AND I <     LEN (A$) THEN X$ = ""
70 B$ = B$ + X$
80  NEXT I
90  PRINT B$

]RUN
CARATTERE DA SOPPRIMERE ?A
STRINGA ?BACAABAA
BCBA
```

Comparazione di una stringa di caratteri

Gli operatori di comparazione sono i seguenti:

= uguale a

< > diverso da

> seguente a

< precedente

La relazione è basata sull'ordine alfabetico, così risulterà

$A < B < C < \ldots.. < Z$

questo poichè le lettere sono codificate, in ASCII, in ordine crescente:

'A' = 65 'B' = 66 ..... 'Z' = 90

Queste comparazioni non si applicano soltanto a caratteri isolati, ma a tutte le stringhe di caratteri.

Esempio:

```
10   INPUT "A$ = ";A$
20   INPUT "B$ = ";B$
30   IF A$ =  < B$ THEN 50
40   PRINT A$;" E' DOPO ";B$: GOTO 10
50   IF A$ = B$ THEN 70
60   PRINT B$;" E' DOPO ";A$: GOTO 10
70   PRINT "IDEM": GOTO 10
```

Conversione di tipo e di codice

Alfanumerica → numerica

VAL: A$ → N

La funzione VAL(A$) consente di estrarre, da una stringa alfanumerica, ogni informazione numerica che comporrà l'inizio della stringa.

Il primo carattere della stringa dev'essere uno degli elementi possibili di un numero (è accettato uno spazio in testa) altrimenti VAL(A$) varrà 0.

In seguito ogni carattere è esaminato fino ad incontrare il primo carattere non numerico (differente da E, +, -, . cifre o spazi). Questo primo carattere incontrato viene ignorato così come vengono ignorati i seguenti, e la stringa viene valutata in numero reale o intero.

Esempio:

```
] PRINT VAL ("2.85E3B1P")

  2850
```

Numerica → alfanumerica

STR$: N → A$

La funzione STR$(N) restituisce la stringa di caratteri (cifre, segni, E, spazi) rappresentato dal valore di N.
Esempio:

```
] PRINT LEN (STR$ (133789))

  6 (numero delle cifre che compongono il numero)
```

Carattere → suo codice ASCII

ASC: A$ → C

La funzione ASC(A$) restituisce il codice ASCII del primo carattere dalla stringa A$ (fare riferimento all'appendice dei codici ASCII dei caratteri della tastiera).

Esempio:

```
] PRINT ASC ("A")

  65

] PRINT ASC ("Ctrl D")

  4
```

Si possono definire nella funzione ASC l'insieme dei caratteri della tastiera (cifre, segni, lettere, caratteri di controllo).

I codici generati sono compresi tra 0 ('*Ctrl*') e 94 ('*SHIFT* N').
Per provare questa funzione:

```
10   GET A$
20   PRINT  ASC (A$)
30   GOTO 10
```

Codice ASCII → carattere corrispondente

CHR$: C → A$

La funzione CHR$(C) restituisce il carattere dove il codice ASCII vale C in numerazione decimale.
Esempio:

] PRINT CHR$(7)

'Beep'

] D$ = CHR$(4) per assegnare il carattere di controllo 'Ctrl-D' alla variabile D$

Caratteri visualizzabili sullo schermo:

C	CHR$(C)
32 - 63	! # Ø % & ' () * +,-./Ø123456789:;<=?
96 - 127	idem
160 - 191	idem
224 - 255	idem
64 - 95	ABCDEFGHIJKLMNOPQRSTUZWXYZ [\] Λ_
192 - 223	idem

Caratteri di controllo:

C	CHR$(C)
0 - 31	CTRL @ABCDEFGHIJKLMNOPQRSTUVWXYZ esc SHIFT M Λ

Caratteri speciali facenti parte dei caratteri di controllo:

C	CHR$(C)
4	Ctrl D (vedere DOS)
7	Ctrl G campanello ("bip")
8	Ctrl H <-- (cancella carattere)
13	Ctrl M Return (ritorno a capo)
15	Ctrl U --> (avanza di una posizione)
18	Ctrl X (annulla una linea)
27	ESC (Escape)

(fare ancora riferimento all'appendice del codice ASCII)

Esercizi

2.5 Da una stringa di caratteri A$, sottrarre la stringa inversa B$ e verificare se è palindroma. Come si produce una stringa palindroma artificiale ?

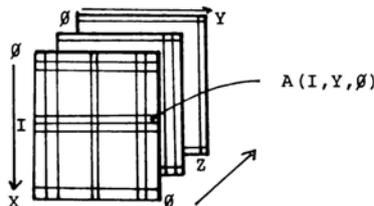
2.6 Visualizzare tutti gli anagrammi (con o senza senso) di una parola di 4 lettere (24).

2.7 Da una stringa di caratteri X$, di lunghezza qualsiasi inferiore a 15, ricostituire una stringa che contenga X$ con uno spazio all'inizio e di lunghezza fissa uguale a 15.

2.8 Visualizzare dei numeri reali con virgola fissa, incolonnati a destra.

LE MATRICI DI VARIABILI

Dimensionamento - Accesso - Occupazione in memoria

L'istruzione DIM A(X,Y,Z) consente di strutturare delle informazioni in una matrice di valori; l'accesso all'informazione è diretta se si conosce la sua posizione nella matrice.

La posizione di una informazione è caratterizzata dai valori interi degli indici.
Esempio:

$$0 \leqslant I \leqslant X$$
$$0 \leqslant J \leqslant Y$$
$$0 \leqslant K \leqslant Z$$

L'informazione stessa è rappresentata da una variabile indicizzata: A(I,J,K).

La quantità di dati inseribile in questa matrice è (X + 1)*(Y + 1)*(Z + 1).

Il numero delle dimensioni è limitata a 88, come DIM A(D1,D2,...D88).

L'insieme delle informazioni di una matrice dev'essere omogenea, vale a dire:

DIM A(X,Y,Z)	per i valori reali
DIM B%(X,Y,Z)	per i valori interi
DIM C$(X,Y,Z)	per le stringhe di caratteri

I valori X, Y, Z sono gli interi che rappresentano i valori massimi degli indici della matrice.

OCCUPAZIONE IN MEMORIA

Le matrici di valori interi o reali occupano la memoria in ragione di 2 byte per i numeri interi e 5 byte per i numeri reali (per la memorizzazione) oltre a 2 byte per i 2 caratteri del nome della matrice, 2 byte per l'indirizzo della matrice seguente, 1 byte per la dimensione della matrice, 2 byte per dimensione contenente il valore massimo dell'indice in quella dimensione.

Una matrice di N dimensioni è dunque identificata e specificata in 5+2*N byte.

A ciascuna delle dimensioni, se si corrisponde $X_{+}1$ valori massimi allora lo spazio occupato, per questi valori, sarà di $(X+1)^N*5$ byte.

Per esempio: DIM A(10,10,10) necessita di 11 byte d'identificazione e 6655 byte di valori reali.

Questo spazio viene riservato dall'istruzione DIM.

Le matrici di variabli alfanumeriche sono memorizzate in ragione di 3 byte di identificazione per ciascuna variabile indice, più 2 byte per il nome della matrice, 2 byte per l'indirizzo della matrice seguente, 1 byte per il numero delle dimensioni, 2 byte per dimensione per memorizzare il valore massimo autorizzato per l'indice di quella dimensione.

Per una matrice di N dimensioni, ciascuna di X+1 stringhe possibili, saranno necessari come minimo $5+2*N+((X+1)^N)*3$ byte.

Le stringhe alfanumeriche stesse non sono state ancora calcolate. Si sa che la lunghezza non può superare i 255 caratteri. Sono memorizzate in una zona che viene loro riservata nella parte alta della memoria, man mano che vengono effettuate le assegnazioni ai loro valori indicizzati.

Viene utilizzata la stessa procedura citata nel paragrafo di pag. 46 (occupazione in memoria di una stringa di caratteri), le stringhe modificate non sono sovrapposte a quelle precedenti, salvo non lo si chieda specificatamente con l'istruzione PRINT FRE(0).

La loro localizzazione e la loro lunghezza sono definite in 3 byte riservati a ciascuna variabile della matrice.

Esempio: DIM A$(100)
occupa un minimo di 7 + 303 = 310 byte d'identificazione di variabile e la restante parte occupata in memoria è calcolata in funzione delle stringhe realmente utilizzate.

L'istruzione DIM è necessaria solo quando si desiderano utilizzare più di 2 dimensioni o più di 11 valori per ogni dimensione.

Una variabile indicizzata T utilizzata in una istruzione, senza che sia stata dimensionata la matrice corrispondente, implica comunque un'area rivervata in memoria equivalente a DIM T(10,10).

Sviluppo di una matrice di dati

Lettura e visualizzazione dei valori memorizzati:

```
5  DIM A(100,4)
10  FOR I = 1 TO 100
20  FOR J = 1 TO 4
30  PRINT A(I,J);" ";
40  NEXT J: PRINT
50  NEXT I
```

Scrittura in memoria dei dati di una matrice

```
5  DIM A(100,4)
10  FOR I = 1 TO 100
20  FOR J = 1 TO 4: PRINT I;" ";J
     ;
30  INPUT A(I,J)
40  NEXT J: NEXT I
```

si sostituisce l'impostazione da tastiera con la lettura automatica dagli statement DATA.

```
30  READ A(I,J)
50  DATA  0,5,10,15,1.6,11,14,...

51  DATA
```

400 dati da inserire nei DATA

ESEMPIO DI UTILIZZAZIONE DELLE ISTRUZIONI READ E DATA

Sviluppo di un questionario

I risultati di un questionario sono dati in ragione di N risposte 0 o 1 per persona intervistata. Ogni persona appartiene a una categoria tra le M possibili.

La matrice TR è strutturata in N linee (1 per domanda) e M colonne (1 per categoria).

Il caso TR(I,J) conterrà il numero delle risposte affermative (1) alla domanda I per le persone appartenenti alla categoria J.

Si suppone che le risposte siamo memorizzate in linee DATA per ogni persona intervistata, nel seguente ordine:
DATA categoria, risposta alla prima domanda, risposta alla seconda domanda, ..., risposta alla N*esima* domanda.

```
5 M = 4:N = 10
10  DIM TR(N,M)
20  READ C: IF C =  - 1 THEN 200
30  FOR Q = 1 TO 10
40  READ R
50 TR(Q,C) = TR(Q,C) + R
60  NEXT Q
```

```
70   GOTO 20
100   DATA 2,0,0,0,1,1,0,1,0,1,0
110   DATA 2,0,1,0,1,0,1,0,0,1,0
120   DATA 3,1,1,1,1,0,0,1,1,0,1
130   DATA 4,1,0,1,1,1,0,0,0,1,0
140   DATA 3,0,1,0,1,1,1,1,1,0,1
150   DATA 1,0,1,0,0,0,0,1,1,0,0
160   DATA 4,1,1,1,0,0,0,1,1,1,1
170   DATA -1
200   REM  VISUALIZZAZIONE DEI RISULTATI
210   FOR Q = 1 TO N
220  TR(Q,0) = Q: REM  PRIMA COLONNA CON I CODICI DELLE DOMANDE
230   FOR C = 0 TO M
240   PRINT TR(Q,C);" ";
250   NEXT C: PRINT
260   NEXT Q

]RUN

1      0      0      1      2
2      1      1      2      1
3      0      0      1      2
4      0      2      2      1
5      0      1      1      1
6      0      1      1      0
7      1      1      2      1
8      1      0      2      1
9      0      2      0      2
10     0      0      2      1
```

Conteggio dei dati alfanumerici

Si consideri un quantitativo di pezzi aventi ciascuno 3 caratteristiche, ciascuna caratteristica può essere di due tipi.

Il programma consente di determinare il numero dei pezzi aventi lo stesso numero di caratteristiche indicate.

```
10   REM  I DATI SONO IN "DATA"
20   REM  PER OGNI PEZZO SONO DATI TRE PARAMETRI
30   REM  O/S, C/P, F/N SONO I DUE STATI
32   REM  DI CIASCUN PARAMETRO
40   FOR I = 1 TO 10: FOR J = 1 TO 3
50   READ D$(I,J)
60   NEXT J: NEXT I
```

```
70  DATA  O,C,F,O,P,N,S,C,F,S,P,N
71  DATA  S,C,N,S,C,F,O,C,N,S,C,F
72  DATA  O,P,F,S,C,N
75 K = 0
80  INPUT C1$,C2$,C3$: IF C1$ = "" THEN  END
90  FOR I = 1 TO 10
100  IF D$(I,1) <  > C1$ THEN 200
110  IF D$(I,2) <  > C2$ THEN 200
120  IF D$(I,3) <  > C3$ THEN 200
150 K = K + 1
200  NEXT I:KM = K
210  PRINT C1$;C2$;C3$;" ";KM
220  GOTO 75

]RUN
?O,P,N
OPN 1
?S,C,F
SCF 3
?S,C,N
SCN 2
```

MEMORIZZAZIONE DI MATRICI DI DATI SU CASSETTA

Una matrice dimensionata con DIM P(X,Y,Z,...) è registrabile su cassetta con l'istruzione STORE P mentre
l'istruzione RECALL P richiama in memoria i valori registrati su cassetta.

I dati non possono essere di tipo alfanumerico.

Non è necessario utilizzare lo stesso nome della matrice per la rilettura dei dati, ma è obbligatorio che la stessa abbia le stesse dimensioni della matrice originale.

Prima di far eseguire le istruzioni di cui sopra nel corso di un programma, è necessario riavvolgere il nastro della cassetta. Un 'Beep' segnala l'inizio della registrazione e un altro 'Beep' ne indica la conclusione.

Durante la registrazione non è visibile il cursore sul video.

Quando la registrazione non termina correttamente solo il tasto RESET permetterà l'interruzione della procedura.

2.9 Un gruppo di 10 allievi supera degli esami in 6 materie. Stampare i voti e le medie per ciascun allievo.

2.10 Con una memoria RAM da 48 Kbyte qual è l'ordine di grandezza maggiore per una matrice di numeri reali ? E di numeri interi ?

2.11 Dato un testo, creare una matrice che fornisca la frequenza di apparizione di ciascuna delle lettere che compongono il testo.

GLI ALGORITMI E LE STRUTTURE DEI DATI

Liste sequenziali (matrice a 1 dimensione).

Metodo lineare

I dati sono inseriti, man mano che compaiono, in coda alla lista, se gli stessi non fanno già parte della lista stessa.

La ricerca di un dato nella lista è fatta sequenzialmente, scandendo gli elementi della lista stessa dall'inizio alla fine.

La soppresione di un dato implica delle operazioni di riscrittura dei dati posti nelle posizioni inferiori.

Esempio:

```
10  REM  ELENCO DEI NOMI
20  DIM LN$(100)
25 N = 0: REM  NUMERO DEGLI ELEMENTI
30  INPUT N$:LN$(N + 1) = N$
40 I = 1
50  IF N$ = LN$(I) AND I = N + 1 THEN N = N + 1: GOTO 30
60  IF N$ = LN$(I) THEN 100
70 I = I + 1: GOTO 50
100  PRINT "E' IL NUMERO ";I
110  INPUT "DESIDERATE CANCELLARLO ? ";R$
120  IF R$ = "SI" THEN 150
130  IF R$ = "NO" THEN 30
140  GOTO 110
150  REM  CANCELLAZIONE
160  FOR J = I TO N - 1
```

```
170 LN$(J) = LN$(J - 1)
180  NEXT J
190 N = N - 1
200  GOTO 30
```

Metodo dicotomico di ricerca

La lista dev'essere precedentemente ordinata (secondo un criterio, per esempio quello alfabetico).

Localizzare il centro della lista e comparare il valore trovato al centro con quello dell'elemento cercato. Se questo valore è troppo grande, localizzare il centro della prima metà e il valore corrispondente e procedere nuovamente alla comparazione di quest'ultimo con il valore cercato. Ripetere la procedura sulla prima metà fintanto che l'elemento cercato non sia stato trovato.

Se il valore del centro è troppo piccolo provare allora con il centro della seconda parte e ripetere la procedura nella seconda metà.

Esempio:

```
10  DIM LN$(100)
20  REM  :L'ELENCO E' GIA' ORDINATO
25  REM  :CONTIENE N ELEMENTI
30  INPUT N$: REM  RICERCA
40 INF = 1:SUP = N
50  IF INF > SUP THEN E = 0: GOTO 120
60 I =  INT ((INF + SUP) / 2)
70  IF N$ < LN$(I) THEN SUP = I + 1: GOTO 50
80  IF N$ > LN$(I) THEN INF = I + 1: GOTO 50
90 E = 1: REM  TROVATO
100  PRINT "E' IL NUMERO ";I;" DELL'ELENCO": GOTO 30
120  PRINT "NON ESISTE !": GOTO 30
```

Classificazione o ordinamento

Un metodo di ordinamento, detto lineare, compara ciascun elemento della lista con tutti quelli seguenti. Quando viene trovato un elemento mal posizionato, questo è scambiato con quello che avrebbe dovuto essere posto prima. Questa procedura si ripeterà alla occorrenza di $N + (N - 1) + (N - 2) + ... + 1 = 1/2N (N - 1)$ comparazioni.

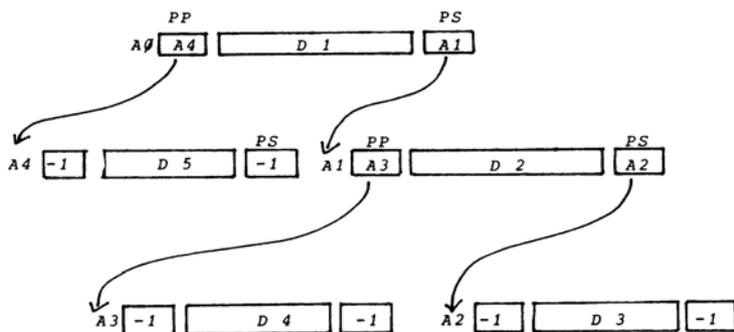
```
100 A = 0:W = 1:J =  - 1
110  IF A =  - 1 THEN R$ = "ASSENTE": RETURN
120  IF X = Z(A) THEN R$ = "PRESENTE": RETURN
130  IF X > Z(A) THEN J = A:A = P(A):W = 1: GOTO 110
140 A =  - 1:W = 0: GOTO 110
200  IF R$ = "PRESENTE" THEN  RETURN
210 Z(I) = X
220  IF W = 1 THEN P(I) =  - 1
230  IF W = 0 THEN P(I) = P(J)
240 P(J) = I
250 I = I + 1
260  RETURN
1000  PRINT "I   Z(I)   P(I)"
1005  FOR I = 0 TO N
1010  PRINT I;"   ";Z(I);"    ";P(I)
1020  NEXT I

]RUN
?0
?200
?346
?20
?345
?347
?94
?20
?1
?1000
```

I	Z(I)	P(I)
0	-1E+38	1
1	0	8
2	200	5
3	346	6
4	20	7
5	345	3
6	347	-1
7	94	2
8	1	4

Il tempo di ricerca, in una lista a concatenazione semplice, è proporzionale al numero degli elementi N, quando una lista è ordinata, la ricerca dicotomica richiederà un tempo proporzionale al logaritmo di N.

Concatenazione doppia

In questo tipo di struttura di dati, si aggiungono due puntatori, uno consente l'accesso al dato seguente, l'altro consente l'accesso al dato precedente. La lista è schematicamente indicata nel seguente disegno:

Se D5< D1< D4< D2< D3

Esempio:

```
10  DIM Z$(100),PP(100),PS(100)
20 Z$(1) = "": REM  STRINGA VUOTA
22 PP(1) =  - 1:PS( + 1) =  - 1: REM  PUNTATORI NON SPECIFICATI
30 I = 1
40  INPUT X$: IF X$ = "STOP" THEN N = I - 1: GOTO 1000
50  GOSUB 100: REM  RICERCA
60  GOSUB 200: REM  INSERIMENTO
70  GOTO 40
100 A = 1:P =  - 1:W = 1
110  IF A =  - 1 THEN R$ = "ASSENTE": RETURN
120  IF X$ = Z$(A) THEN R$ = "PRESENTE": RETURN
130 P = A: REM  INDICE DELL'ULTIMO TEST
```

```
150  IF X$ > Z$(A) THEN A = PS(A):W = 1: GOTO 110
160 A = PP(A):W = 0: GOTO 110
200  IF R$ = "PRESENTE" THEN  RETURN
210  IF W = 0 THEN PP(P) = I
220  IF W = 1 THEN PS(P) = I
230 P = I: REM  I=PRIMO INDIRIZZO DISPONIBILE
240 I = I + 1
250 Z$(P) = X$
260 PP(P) =  - 1
270 PS(P) =  - 1
300  RETURN
1000  REM  VISUALIZZAZIONE
```

DATI MULTIPLI E MEMORIZZAZIONE

Se si dispone di un insieme di memorizzazioni o ciascuna memorizzazione è composta da molte informazioni caratteristiche, per realizzare su queste memorizzazioni delle operazioni di ordinamento, si eviteranno scambi ripetitivi per ogni caratteristica utilizzando una tabella di puntatori sulla quale gli scambi saranno realizzati virtualmente. L'ordinamento della lista sarà realizzato da un accesso indiretto, la lista non sarà modificata.

Esempio:

Si consideri una lista di allievi di una classe e le votazioni loro attribuite. Si desidera visualizzare la loro classificazione, vale a dire l'elenco dei nomi per ordine decrescente in relazione ai voti ottenuti.

La variabile N(I) dei voti è quella sulla quale dev'essere effettuato l'ordinamento. I voti sono riscritti in NP.

Si costruisce una matrice ausiliaria di puntatori P(1), P(2)...P(N) tale che P(1) sia l'indice dell'allievo che dev'essere classificato primo. Il nome di questo allievo sarà dunque N$(P(1)).

Si ricercano, successivamente i migliori voti della lista NP e si sostituiscono i voti già scelti con il valore -1 che sarà sempre inferiore a tutti gli altri.

```
10   DIM N$(30),N(30),P(30),NP(30)
20   FOR I = 1 TO 30
30   READ N$(I),N(I):NP(I) = N(I)
40   NEXT I
60   FOR I = 1 TO 30
70 J = 1:M = NP(J)
80   FOR K = 2 TO 30
90   IF NP(K) <  = M THEN 100
95 J = K:M = NP(K)
100  NEXT K
110 P(I) = J:NP(J) =  - 1
120  NEXT I
150  FOR I = 1 TO 30
160  PRINT I,N$(P(I)),N(P(I))
170  NEXT I
200  DATA  PIERO,9,PAOLO,9.75,.....
```

Gli algoritmi di ordinamento esposti a pag. 62, 63 sono sempre utilizzabili se i dati sono multipli, sarà necessario, per ciascuna istruzione di scambio, in parallelo, aggiungere le istruzioni di modifica dell'elenco dei puntatori.

In precedenza, è necessario dimensionare questo elenco e inizializzarlo con valori d'indice crescenti

```
DIM P(101)
FOR I = 1 TO 101:P(I) = I: NEXT I
```

Per il *Quik Sort* ecco di seguito le modifiche:

```
100 I = LI:J = LS:TEMP = T(I):PT = P(I)
130  IF J <  = I THEN T(I) = TEMP:P(I) = PT: GOTO 200
140 T(I) = T(J):P(I) = P(J):I = I + 1
170  IF J > I THEN T(J) = T(I):P(J) = P(I):J = J - 1: GOTO 110
180 T(J) = TEMP:P(J) = PT:I = J
```

FUNZIONI DIVERSE

Si tratta di funzioni non matematiche che facilitano la stampa o la visualizzazione dei risultati.

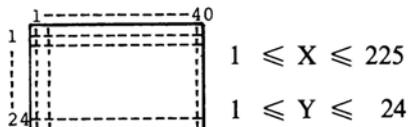
Tabulazione assoluta

Prima dell'istruzione PRINT, il cursore può essere posizionato in qualunque punto dello schermo con le istruzioni:

HTAB X in orizzontale

VTAB Y in verticale

$1 \leqslant X \leqslant 225$

$1 \leqslant Y \leqslant 24$

La linea superiore dello schermo corrisponde a Y = 1

L'istruzione HTAB posiziona il cursore partendo dal margine sinistro e dalla parte alta dello schermo e consente l'esplorazione, linea per linea, di tutto il video.

La posizione orizzontale sarà data da

((X—1)MOD 40)+1

Esempio:

Visualizzazione di un risultato numerico in una posizione prefissata sullo schermo.

```
10  FOR I = 99 TO 0 STEP  - 4
20  HTAB 20: VTAB 12
30  PRINT I;" ";
40  NEXT I
```

Passando dai numeri con 2 cifre a quelli di una sola cifra, e' necessario prevedere la visualizzazione del carattere piu' a destra sullo schermo.

Tabulazione relativa

In una istruzione PRINT, il cursore può essere spostato di un certo numero di spazi tra due valori da visualizzare.

Questo è l'oggetto dell'istruzione SPC(X) che dovrà essere racchiusa tra punto e virgola nell'istruzione PRINT.

$0 \leqslant X \leqslant 255$

Dunque SPC(X) sposta il cursore di X caratteri partendo dall'ultimo carattere visualizzato.

L'istruzione TAB(Z)

Nell'istruzione PRINT, il cursore può essere posizionato ad una distanza fissa dal margine sinistro con l'istruzione TAB(Z).

$$0 \leqslant Z \leqslant 255$$

Se la posizione attuale del cursore è superiore a Z questo non sarà spostato.

La posizione del cursore può essere visualizzata con l'istruzione PRINT aiutata dall'istruzione POS(0).

Il valore ottenuto parte da 0 se il cursore è sul margine sinistro.

Esempio:

```
PRINT TAB (23); POS (0)
22
PRINT SPC (23); POS (0)
23
```

Finestra di schermo

La visualizzazione standard, e le sue funzioni di tabulazione, controllano il cursore e lo spostano automaticamente di una posizione a destra dopo che un carattere è stato visualizzato, lo pone alla prima posizione della linea seguente quando sorpassa il margine destro e sposta di una linea verso l'alto tutto il testo visualizzato quando il cursore sorpassa l'ultima linea di schermo ponendolo alla prima posizione della linea in basso.
Tutte queste operazioni automatiche possono essere eseguite su un formato di schermo ridotto (finestra).

Si definisce una "finestra" di schermo al di fuori della quale non si troverà mai il cursore che passerà automaticamente alla linea seguente, oppure il testo visualizzato sarà spostato verso l'alto, per liberare una linea vuota, ogni volta che saranno raggiunti i limiti della finestra.

Quattro parametri caratterizzano la finestra:

- il margine sinistro	$0 \leqslant L \leqslant 39$
- la larghezza	$1 \leqslant W \leqslant 40$
- l'alto	$0 \leqslant T \leqslant 23$
- il basso	$0 \leqslant B \leqslant 24$

Attenzione: $L + W < 40$
$T < B$

Valori standard:

L = 0
W = 40
T = 0
B = 24

Indirizzo dei parametri e istruzioni di modifica degli stessi:

```
POKE  32,L
POKE  33,W
POKE  34,T
POKE  35,B
```

Indirizzi delle posizioni del cursore sullo schermo:

CH = PEEK(36)	: attuale posizione orizzontale
POKE 36,CH	: spostamento orizzontale del cursore
CV = PEEK(37)	: attuale posizione verticale
POKE 37,CV	: spostamento verticale del cursore
$0 \leqslant CV \leqslant 23$	: posizione assoluta sullo schermo
$0 \leqslant CH \leqslant W$	: posizione relativa al margine sinistro della finestra.

Cancellazione del testo

Chiamando dei sotto-programmi del Monitor, si possono ottenere delle modifiche al testo visualizzato sullo schermo.

CALL -936: equivale all'istruzione HOME. Il cursore è posizionato in alto a sinistra dello schermo. Lo schermo è completamente cancellato (nei limiti della finestra).

CALL -958: equivale a ESC F. Cancella tutti i caratteri dopo la posizione attuale del cursore fino al termine della pagina.

CALL -868: equivale a ESC E. Cancella tutti i caratteri dopo la posizione attuale del cursore fino al termine della linea.

Salto di linea e spostamento verso l'alto

CALL -922: equivale a CTRL-J o PRINT. Salto di una linea

CALL -912: spostamento verso l'alto, di una linea, di tutte le linee di testo all'interno della finestra di schermo.
La linea più in alto è persa e rimpiazzata da quella che segue. La linea più in basso è libera.

FUNZIONI MATEMATICHE

In un linguaggio evoluto com'è l'Applesoft, sono disponibili numerose funzioni matematiche che si possono utilizzare con molta semplicità, facendo attenzione a dare come argomento solo valori che siano riconosciuti dalla funzione.

Valori interi - Valori assoluti - Segni

Y = INT(X) restituisce il più grande numero intero inferiore o uguale a X

Y = ABS(X) restituisce il valore assoluto di X, vale a dire:

$Y = X$ se $X \geqslant 0$
$Y = -X$ se $X < 0$

Y = SGN(X) restituisce il segno del valore X, vale a dire:

$Y = -1$ se $X < 0$
$Y = 0$ se $X = 0$
$Y = +1$ se $X > 0$

Generazione di numeri casuali

Y = RND(1) fornisce a Y un valore reale compreso tra 0 e 1 (1 escluso). Questo valore è imprevedibile e può essere considerato 'tirato a caso'; questo valore è chiamato *numero casuale*.

Il sistema che elabora questi numeri casuali è un generatore basato sul numero ottenuto incrementando due locazioni di memoria ($4E e $4F), infatti l'utilizzatore può battere un tasto della tastiera solo durante un'istruzione INPUT o un'istruzione GET.

Questo generatore di numeri casuali può essere utilizzato per ottenere sempre la stessa sequenza di numeri casuali.

Programmando Y = RND(-1), il numero generato è sempre lo stesso, e se si chiama RND(1) per generare una serie di numeri casuali, questa serie sarà sempre la stessa.

Esempio:

```
10  PRINT  RND ( - 1)
20  PRINT  RND (1)
30  GOTO 20
```

Questo programma genera sempre la medesima sequenza.

Per cambiare la sequenza è necessario modificare l'argomento negativo dell'istruzione nella linea 10.

Y = RND(0) restituisce l'ultimo numero casuale generato nel corso del programma.

Applicazione della funzione RND

Per generare un numero intero E, compreso tra MAX e MIN, si normalizza la funzione RND nel seguente modo:

E = INT [(MAX—MIN+1)*RND(1)+MIN]

Esempio:

Sorteggio di una lettera dell'alfabeto

I codici ASCII dell'alfabeto si intendono da 65 ('A') a 90 ('Z')
L$ = CHR$(INT(26*RND(1)+65))

Generazione automatica di un valore casuale 0 o 1

```
10 V% = 2 *  RND (1)
20  PRINT V%;
30  GOTO 10
```

FUNZIONI ESPONENZIALI E LOGARITMICHE

Y = SQR(X): estrae la radice quadrata di X più rapidamente che con l'espressione

$Y = X \wedge 0.5$

Y = EXP(X): eleva e = 2.718289 alla potenza di X.

Y = LOG(X): restituisce il logaritmo naturale (base e) di X.

Esempio:

```
PRINT EXP(0),LOG(EXP(1))

  1          ,1
```

FUNZIONI TRIGONOMETRICHE

Le funzioni sono calcolate in Applesoft supponendo che l'argomento sia dato in radianti.

```
PRINT SIN (3.14159/2), SIN (3*3.14159/2)
1                     ,—1
PRINT COS (2 * 3.14), TAN (3.14/4)
1                     .999998674
```

È disponibile anche la funzione inversa:

ATN (X) restituisce un valore angolare espresso in radianti, compreso tra — 3.14/2 e + 3.14/2.

FUNZIONI PREDEFINITE

Se un calcolo dev'essere ripetuto sovente in un programma, ogni volta con un argomento differente, è vantaggioso definire la funzione riguardante il calcolo con:
DEF FN A(X) = X + 17.5 * X/100
in testa al programma, utilizzandola poi come una classica funzione matematica. Il valore attuale di X sarà sostituito nell'espressione e il risultato sarà restituito con Y = FNA(X).

LE ISTRUZIONI PEEK, POKE E CALL

Un programma scritto in Applesoft può comunicare con il sistema Apple (il Monitor, le funzioni di entrata/uscita), accedendo a degli indirizzi assoluti in memoria sia in lettura che in scrittura:

POKE A,V: scrive il valore V (espresso in base 10) nella locazione di memoria d'indirizzo assoluto A (espresso in base 10).

X = PEEK(A): restituisce, nella variabile X il contenuto della locazione di memoria d'indirizzo assoluto A (espresso in base 10 o eventualmente per complemento a 65536).

Ciascuna locazione di memoria è costituita da 8 bit (1 byte), ciò limita a $2^8 — 1 =$ 255 il massimo valore che può essere scritto in un byte. Dunque $0 \leqslant V \leqslant 255$.

L'indirizzo assoluto A appartiene a tutto lo spazio indirizzabile dal microprocessore che s'intende da 0 a 64 Kbyte (65535) o da $0 a $FFFF.

Si fà riferimento ad un indirizzo particolare A con un numero intero positivo $0 < A < 65535$ oppure con un numero negativo A' tale che A' = — (65536—A) se l'indirizzo cercato è superiore a 32767.

CALL A: rappresenta una chiamata ad un sotto-programma scritto in linguaggio macchina (generalmente incluso nel Monitor) che inizia all'indirizzo assoluto A in memoria centrale (RAM o ROM).

Esempio:

```
] CALL —151
*
```

A = —151 oppure $FF69 è l'indirizzo d'ingresso del programma Monitor.

```
] PRINT PEEK (A+1)*256 + PEEK(A)
```

visualizzerà il contenuto (in base 10) di una parte di memoria adiacente rappresentando un valore su 16 bit.
L'indirizzo A contiene gli 8 bit meno significativi, l'indirizzo A+1 contiene gli 8 bit più significativi.

Cerchiamo il valore della parte più alta della memoria RAM disponibile. E' contenuto negli indirizzi $73 e $74.

```
] PRINT PEEK(116)*256 + PEEK(115)
38400
```

Spostamento di una zona di memoria

Il sotto-programma che sarà utilizzato è disponibile nel programma Monitor in $FE2C e si chiama con CALL —468.

I parametri chiesti dal sotto-programma comprendono:

- l'indirizzo d'inizio della zona da trasferire,
- l'indirizzo finale della zona da trasferire,
- l'indirizzo di destinazione.

Questi tre parametri sono caricati,prima della chiamata, negli indirizzi $3C, $3D, $3F, $42, $43.

Esempio:

$400 - $7FF da trasferire in $800

```
10   POKE 60,0: POKE 61,4
20   POKE 62,255: POKE 63,7                 $400 -> $3C,$3D
30   POKE 66,0: POKE 67,8                   $7FF -> $3E,$3F
40   CALL  - 468                            $800 -> $42,$43
```

Memorizzazione di stringhe di caratteri su cassetta

Il sotto-programma di registrazione di dati su cassette inizia in $FECD, si chiama con CALL -259.

Il sotto-programma di lettura dati da cassetta inizia in $FEFD, si chiama con CALL -307.

I parametri chiesti dal sotto-programma sono:
- indirizzo di inizio della zona di memoria,
- indirizzo finale della zona di memoria
da registrare o da leggere.

Questi parametri sono caricati negli indirizzi $3C, $3D e $3E, $3F.

La lunghezza della zona, che contiene le stringhe, è determinata dalla differenza tra HIMEM e DC contenuti in $73, $74 e $6F, $70.

Questa lunghezza, dopo il calcolo, sarà registrata per prima sulla cassetta e posizionata, temporaneamente, in $30 e $31.

```
10  REM  SALVATAGGIO DI STRINGHE DI CARATTERI
15  REM  SU CASSETTA
20  DIM A$(10)
30  FOR K = 1 TO 9: INPUT A$(K): NEXT K
40  GOSUB 1000
50  PRINT "LE STRINGHE SONO REGISTRATE."
60  PRINT "PER RICARICARLE IN RAM"
70  PRINT "IMPOSTARE GOTO 100, RIAVVOLGETE IL NASTRO"
80  PRINT "AVVIARE LA CASSETTA POI PREMERE RETURN"
90  END
100  REM  CARICAMENTO IN MEMORIA
110  DIM B$(10)
120  GOSUB 2000
130  FOR K = 1 TO 9: PRINT B$(K): NEXT K
140  END
1000  REM  SALVATAGGIO DI A$
1005  PRINT "PREPARARE LA REGISTRAZIONE"
1010  PRINT "COMINCIARE LA REGISTRAZIONE"
1015  GET Z$
1020 PPX =  FRE (0): STORE A$: REM  REGISTRATI I VERI PUNTATORI DI A$
1025 XH =  PEEK (115) +  PEEK (116) * 256: REM  HIMEM
1030 XDC =  PEEK (111) +  PEEK (112) * 256: REM  DC
1040 XL = XH - XDC: GOSUB 2100
1050  POKE 30,XL -  INT (XL / 256) * 256
1052  POKE 31,XL / 256
1055  CALL  - 307: REM  REGISTRAZIONE DI XL
1060  POKE 60, PEEK (111): POKE 61, PEEK (112)
1062  POKE 62, PEEK (115): POKE 63, PEEK (116)
1065  CALL  - 307: REM  REGISTRAZIONE DELLE STRINGHE
1070  PRINT "REGISTRATO !": RETURN
2000  RECALL B$: REM  CARICAMENTO DEI PUNTATORI
2010  GOSUB 2100: CALL  - 259: REM  LUNGHEZZA
2020 X =  PEEK (30) +  PEEK (31) * 256
2030 X =  PEEK (115) +  PEEK (116) * 256 - X
2040  POKE 60,X -  INT (X / 256) * 256
2042  POKE 61,X / 256
2044  POKE 62, PEEK (115)
2045  POKE 63, PEEK (116)
2050  CALL  - 259: REM  LETTURA STRINGHE
2060  RETURN
2100  POKE 60,30: POKE 61,0: POKE 62,31: POKE 63,0
2110  RETURN
```

CAPITOLO 3

IL DISEGNO E LA GRAFICA

La visualizzazione sullo schermo catodico (video) è organizzata, nel sistema Apple, secondo tre metodi distinti (caratteri, quadratini, punti), è necessario l'utilizzo di zone della memoria RAM (4 zone a scelta) dove sono memorizzate le informazioni da visualizzare.

Descriveremo come si restringe o si allarga la trama che riceverà i grafici per la visualizzazione sullo schermo.

Una volta scelto il modo, la risoluzione e la zona di memoria che caratterizzano l'immagine da generare, l'utilizzatore non deve far altro che definire, in un sistema di assi cartesiani, quali sono i punti, le rette, le linee spezzate che si vogliono disegnare sullo schermo, precisando con quale colore queste figure dovranno essere visualizzate.

Un modo più complesso è a nostra disposizione per rappresentare delle figure: esso consiste nel pre-codificare uno o più contorni di figure, e domandare che queste figure siano disegnate automaticamente in una scala determinata, con direzione variabile partendo da un punto qualsiasi dello schermo.

Sarà necessario lavorare con il programma Monitor, per pre-codificare queste figure, ma poi Applesoft domanderà solo il loro numero d'ordine, la scala e la posizione per il tracciamento.

I 'MODI' GRAFICI

Modo alta risoluzione

La trama (virtuale) dello schermo è un reticolo di 192 linee orizzontali raggruppate in 24 (grosse) linee ciascuna costituita di 8 linee elementari.

Ogni linea visualizza tutto, o parte, del contenuto di 40 byte di memoria RAM.

Ad ogni punto, di una linea, corrisponde il bit di un byte del quale solo 7 bit sono visualizzati.

Questo porta ad una risoluzione orizzontale di 40 x 7 = 280 punti.

Così dunque la trama di base consente la visualizzazione di:

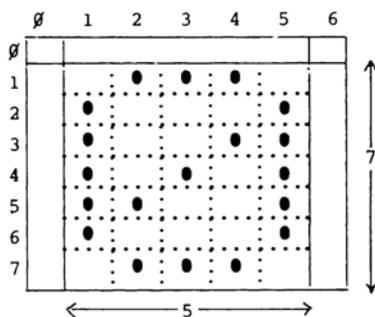
192 x 280 = 53 760 punti distinti

Questa massima precisione possibile dell'immagine è chiamata "grafica alta risoluzione", e corrisponde al modo HGR.

Per l'utilizzatore, nel modo grafico HGR, lo schermo è diviso in:

280 colonne e 192 linee

Modo testo

Il modo alta risoluzione è utilizzato per il generatore di caratteri che costruisce un carattere sotto forma di matrice di punti nel formato 5 x 7, per esempio, la cifra 0 è disegnata sullo schermo:

A destra e a sinistra di ciascun carattere è lasciata libera una colonna, sopra è lasciata libera una linea.

Il modo di tracciare i caratteri è chiamato 'modo testo', e per l'utilizzatore che vuole visualizzare dei caratteri la risoluzione diventa:

280 / 7 = 40 caratteri per linea
192 / 8 = 24 linee di caratteri

Ogni byte di memoria codificato in ASCII è decodificato dal generatore di caratteri (i 6 bit più a destra).

Modo bassa risoluzione

Il terzo modo di tracciamento corrisponde ad una risoluzione meno fine dell'immagine, ma consente di ottenere un set di colori più esteso.

Si chiama "bassa risoluzione" o GR. Infatti i punti sono raggruppati in una matrice 7 x 4 per costituire un quadratino. Al posto di un carattere, nel modo testo, si potranno visualizzare due quadratini sovrapposti.

	ø	1	2	3	4	5	6
ø	.		.		.		.
1	.		.		.		.
2	.		.		.		.
3	.		.		.		.
4		.		.		.	
5		.		.		.	
6		.		.		.	
7		.		.		.	

La sovrapposizione di punti con quattro colori differenti può produrre un colore, tra i 16 possibili (tra cui il nero), per ciascun quadratino.

In bassa risoluzione, detto modo GR, lo schermo è diviso in:

280 / 7 = 40 quadratini per linea orizzontale
192 / 4 = 48 quadratini per colonna

In questo 'modo', un byte in memoria caratterizza due quadratini sovrapposti, i 4 bit più significativi contengono il codice del colore desiderato per il quadratino inferiore e i 4 bit meno significativi contengono il codice colore del quadratino superiore.

Parte della zona di memoria utilizzata per lo schermo secondo il modo di visualizzazione.

HGR	40	byte per linea o 280 punti per linea
	x 192	linee
	= 7860	byte rappresentano l'immagine dello schermo.
TEXT	40	caratteri per linea
	x 24	linee
	= 960	byte di memoria per la visualizzazione 'testo'.
GR	40	quadratini per linea
	x 48	quadratini per colonna
	1920	quadratini, 2 per ogni byte
	= 960	byte di memoria per la visualizzazione GR.

In pratica, le zone riservate sono più ampie: HGR occupa 8 Kbyte, TEXT e GR 1 Kbyte.

Queste zone riservate dispongono dunque di qualche locazione non utilizzata per i grafici, ma qualche volta utilizzate, dal sistema Apple, come memorie temporanee ("Scratch-pad" memorie I/O).

LE ZONE DI MEMORIA RAM RISERVATE AI GRAFICI

In modo alta risoluzione

Le informazioni trasmesse allo schermo video provengono da una delle due pagine, o zone di memoria, seguenti:

pagina n.1: (HGR)

Indirizzo d'inizio:	$2000 o 8192
Lunghezza:	$1FFF o 8192 byte
Indirizzo finale:	$3FFF o 16383

pagina n.2: (HGR2)

Indirizzo d'inizio:	$4000 o 16384
Lunghezza:	$1FFF o 8192 byte
Indirizzo finale:	$5FFF o 24575

Per collegare la pagina 1 allo schermo, partendo dal modo testo, è necessario impostare HGR. Per collegare la pagina 2 allo schermo, partendo dal modo testo, è necessario impostare HGR2.

Per la scelta di una delle due pagine, o eventualmente delle due pagine grafiche, è necessario conoscere sia la configurazione del sistema Apple utilizzato (per utilizzare la pagina 2 è necessario disporre di almeno 24 Kbyte di RAM), sia la dimensione del programma per quanto concerne la zona ad esso riservata, la zona dei dati numerici e la zona dei dati alfanumerici.

Cenni sulle zone d'implementazione di un programma in Applesoft:

Normalmente il caricamento di un programma si fà partendo dall'indirizzo $801. Questo indirizzo si trova negli indirizzi $67 e $68, e può essere modificato cambiando il contenuto di queste locazioni.

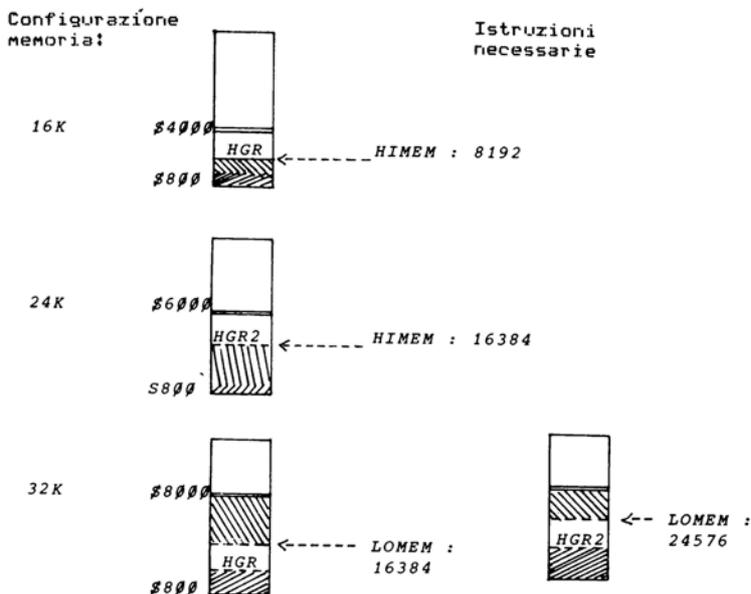
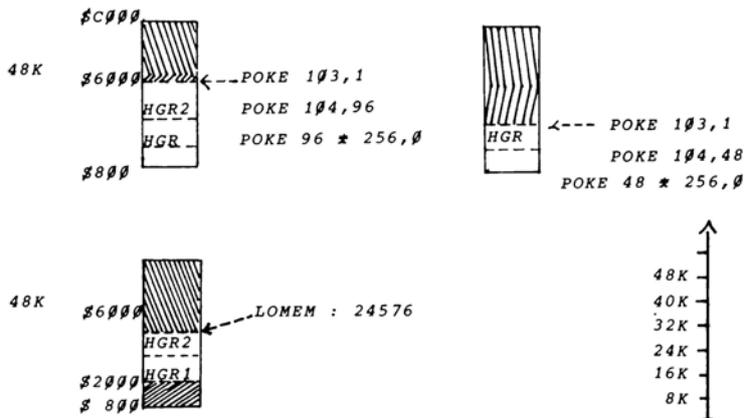
Per esempio, volendo caricare un programma in $C00, è necessario far prima eseguire le seguenti istruzioni:

POKE 103,1:POKE 104,12:POKE 3072,0

Dopo la zona programma (istruzioni interpretate e codificate) normalmente si trova la zona dei dati numerici e gli indirizzi dei dati alfanumerici.

L'inizio di questa zona è una variabile chiamata LOMEM (LOw MEMory). Il suo valore è posto nelle locazioni $69 e $6A e può essere modificata con l'istruzione LOMEM: xxx dove xxx è un indirizzo espresso in numerazione decimale dal quale partirà poi la zona dati.

I valori delle variabili alfanumeriche occupano la parte più alta della memoria RAM senza oltrepassarne il 'tetto' o l'indirizzo chiamato HIMEM (HIgh MEMory). Il suo valore è posto negli indirizzi $73 e $74 e può essere modificata con l'istruzione HIMEM: zzz dove zzz è il valore decimale dell'ultima locazione disponibile in RAM, indirizzo oltre il quale saranno obbligatoriamente posti i dati e il programma.

Qualche soluzione per rendere assolutamente indipendenti le zone per la grafica ad alta risoluzione e le zone occupate da dati e programmi:

I tratteggi obliqui verso destra corrispondono allo spazio riservato alla zona programma.

I tratteggi obliqui verso sinistra corrispondono allo spazio riservato ai dati numerici e alfanumerici.

Le soluzioni sopra proposte evitano tutti i rischi derivanti dalle interferenze parassite tra un programma e le immagini prodotte dallo stesso.

Questo meccanismo di protezione della zona occupata da un programma si ritrova quando un programma Applesoft si serve di un sotto-programma in linguaggio macchina implementato in RAM dall'utilizzatore.

La bassa risoluzione e il modo testo

Le informazioni grafiche inviate allo schermo giungono da una delle due pagine o dalle seguenti zone:

Pagina n.1 (TEXT o GR)

Indirizzo d'inizio:	$400 o 1024
Lunghezza:	$3FF o 1024 byte
Indirizzo finale:	$7FF o 2047

Pagina n.2

Indirizzo d'inizio:	$800 o 2048
Lunghezza:	$3FF o 1024 byte
Indirizzo finale:	$BFF o 3071

L'istruzione TEXT assicura la connessione tra la pagina 1 e lo schermo, e a partire da questa condizione, l'istruzione GR comanda la visualizzazione di grafici in bassa risoluzione sulla pagina 1.

Per la pagina 2, le operazioni da realizzare sono più complesse, saranno spiegate più avanti.

Si noti che la pagina 1 ($400 - $7FF) è sempre disgiunta dalla zona d'implementazione di un programma scritto in Applesoft.

CAMBIAMENTO DEI VARI MODI

I cambiamenti o commutazioni dei modi e della pagina sono realizzati con l'aiuto di 8 specifici indirizzi che, quando sono chiamati, provocano la selezione, tra le 10 combinazioni possibili, di un tipo di visualizzazione.

Questi indirizzi sono raggruppati per 2, quando un indirizzo è posto a 1 (attivo) automaticamente l'altro è posto a 0 (disattivo), e il modo corrispondente viene attivato o disattivato.

Indirizzi di cambio di modo	Effetti corrispondenti:
$CØ5Ø	Modo grafico
$CØ51	Modo testo
$CØ52	Modo non misto
$CØ53	Modo misto grafico ridotto (40 linee) e testo (4 ultime linee dello schermo)
$CØ54	Visualizzazione della pagina 1
$CØ55	Visualizzazione della pagina 2
$CØ56	Grafica a bassa risoluzione
$CØ57	Grafica a alta risoluzione

Per ottenere un cambio di modo in un programma Applesoft, è necessario utilizzare un'istruzione POKE A,0 dove A è il valore decimale di uno degli indirizzi di cui sopra.

Il valore A può essere dato in una delle due forme sotto riportate:

- numero intero positivo A
- numero intero negativo uguale al complemento di 65536 secondo la relazione: —(65536 — A).

Conversioni esadecimali - decimali

ØCØ5Ø	ØCØ51	ØCØ52	ØCØ53	ØCØ54	ØCØ55	ØCØ56	ØCØ57
49232	49233	49234	49235	49236	49237	49238	49239
-16304	-16303	-16302	-16301	-16300	-16299	-16298	-16297
Grafica	Testo	Non misto	Misto	1 Numero delle pagine	2	bassa risoluzione	alta

Effetti dei Poke Grafici

Stato iniziale	Comando	Stato finale
Modo grafico bassa risoluzione	POKE -16297,Ø	Modo grafico alta risoluzione
Modo grafico alta risoluzione	POKE -16298,Ø	Modo grafico bassa risoluzione
Modo grafico	POKE - 163Ø3,Ø	Modo testo
Pagina 1	POKE - 16299,Ø	Pagina 2
Pagina 2	POKE - 163ØØ,Ø	Pagina 1
Modo misto Immagine + 4 linee di testo	POKE - 163Ø2,Ø	Grafica su tutto lo schermo
Grafica su tutto lo schermo	POKE - 163Ø1,Ø	Grafica + 4 linee di testo nella parte bassa dello schermo

Effetti delle istruzione grafiche standard
sui cambiamenti di modo

Stato iniziale	Istruzione	Stato finale
Modo testo	GR	Modo grafico misto a bassa risoluzione + 4 linee di testo nella parte bassa dello schermo. Visualizzazione e cancellazione della pagina 1.
Modo grafico	TESTO	Modo testo. 40 caratteri per linea - 24 linee. Visualizzazione della pagina 1.
Modo testo	HGR	Modo grafico misto ad alta risoluzione + 4 linee di testo. Visualizzazione e cancellazione della pagina 1
Modo testo	HGR 2	Modo grafico alta risoluzione 280 x 192 punti. Visualizzazione e cancellazione della pagina 2

Esempio:

Visualizzazione della pagina 2 in modo testo:

```
] TEXT
] POKE — 16299,0
```

ritorna la pagina 1 in modo TEXT:

```
] POKE — 16300,0
```

Questo comando, al momento della sua scrittura, non appare sullo schermo, poichè lo schermo è connesso alla pagina 2. Quando il comando è stato eseguito, si torna alla pagina 1 e si constata che la scrittura del comando è stata visualizzata su questa pagina.

Visualizzazione di un'immagine grafica ad alta risoluzione in pagina 1, in modo non misto:

```
] TEXT
] HGR
] POKE — 16302,0
```

ritorna al modo misto:

```
] POKE — 16301,0
```

ritorna al testo su tutto lo schermo:

```
] TEXT
```

In modo "MONITOR", visualizzazione dell'immagine grafica in bassa risoluzione non mista in pagina 1 ($400 - $3FF).

```
*C054 "Return"
*C056 "Return"
*C052 "Return"
*C050 "Return"
```

Rappresentazione colorata della zona $800 - $BFF (pagina 2 - bassa risoluzione):

```
*C055
*C056
*C052
*C050
```

In Applesoft:

```
] TEXT
] POKE - 16302,0      (non misto)
] POKE - 16290,0      (bassa risoluzione)
] POKE - 16290,0      (pagina 2)
] POKE - 16304,0      (grafici)
```

L'utilizzo della seconda pagina nel modo testo o grafico in bassa risoluzione obbliga a scalare l'indirizzo di caricamento di un programma Applesoft da $801 a $C00 per preservare la pagina grafica o di testo (da $800 a $BFF).

La scrittura di un testo su questa pagina si può fare soltanto trasferendo un testo composto nella pagina 1 in pagina 2 per mezzo di un programma assemblatore che corrisponde al comando Monitor:

```
* 800 < 400.3FF M
```

LE FUNZIONI GRAFICHE

Il modo bassa risoluzione

I più piccoli elementi visualizzabili sullo schermo sono dei rettangoli fatti in modo che lo stesso ne possa contenere, al massimo, 40 in orizzontale e 48 in verticale.

Il colore di ogni elemento è programmabile con l'istruzione COLOR = numero del codice relativo al colore.

I codici dei colori sono compresi tra 0 e 15, e la corrispondenza tra numero di codice e colore è verificabile facendo eseguire il programma

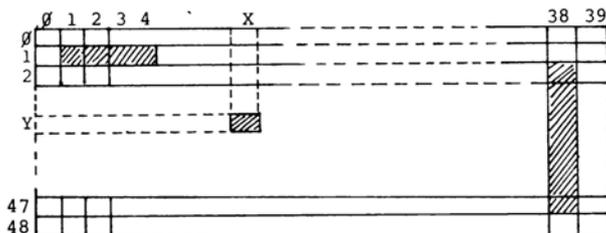
DEMO COLOR APPLESOFT

che fornisce i codici dei colori.

Per visualizzare un quadratino sullo schermo, è necessario fare eseguire l'istruzione PLOT X,Y dopo aver fatto eseguire l'istruzione COLOR = N il tutto dopo aver impostato GR.

Il sistema di coordinate in bassa risoluzione è definita in orizzontale con X o ascissa, vale a dire la distanza che separa il quadratino dal bordo sinistro, e in verticale con Y o ordinata, vale a dire la distanza che separa il quadratino dalla parte superiore dello schermo.

L'ascissa X è compresa tra 0 e 39.

L'ordinata Y è compresa tra 0 e 48 in modo non misto, e tra 0 e 39 in modo misto, dove le quattro linee inferiori sono utilizzate per la visualizzazione di caratteri.

Ogni elemento del video in bassa risoluzione può essere testato con la funzione SCRN (X,Y) che restituisce il numero di codice del colore dell'elemento posto in X,

Sono disponibili delle funzioni per tracciare direttamente delle linee orizzontali e verticali:

HLIN X1, X2 AT Y:
traccia una linea orizzontale d'ordinata Y tra le colonne X1 e X2.

VLIN Y1, Y2 AT X:
traccia una linea verticale d'ascissa X tra le linee Y1 e Y2.

Sulla figura della pagina precedente sono state tracciate:

HLIN 1, 4 AT 1 e
VLIN 2, 47 AT 38

Esempi:

Tracciare un quadrato nel colore I nel modo misto in bassa risoluzione:

```
10  GR
20  COLOR= I
30  FOR K = 0 TO 39
40  PLOT K,0: REM  ALTO
50  PLOT K,39: REM  BASSO
60  PLOT 0,K: REM  ANGOLO SINISTRO
70  PLOT 39,K: REM  ANGOLO DESTRO
80  NEXT K
```

Animazione di un quadrato con successivi spostamenti dei colori in senso orario e antiorario:

```
5  DIM XA(40),YA(40),XN(40),YN(40)
6  REM  DEFINIZIONE DEL PERCORSO
10  FOR M = 0 TO 9
12 XA(M) = M:YA(M) = 0: REM             XA,YA
13 XN(M) = XA(M) + 1:YN(M) = 0: REM     POSIZIONE
14  NEXT M: REM                         ATTUALE
```

```
15  FOR M = 10 TO 19: REM             XA,YA
16 XA(M) = 10:YA(M) = M - 10: REM     POSIZIONE
17 XN(M) = 10:YN(M) = YA(M) + 1: REM  SEGUENTE
18  NEXT M

20  FOR M = 20 TO 29
22 XA(M) = 30 - M:YA(M) = 10
23 XN(M) = XA(M) - 1:XN(M) = 10
24  NEXT M

25  FOR M = 30 TO 39
26 XA(M) = 0:YA(M) = 40 - M
28 XN(M) = 0:YN(M) = YA(M) - 1
30  NEXT M
35  GR
40 K =  - 1
41  REM  REGISTRAZIONE DEI COLORI IN SERIE
42  INPUT C0:K = K + 1
44  IF K < 1 THEN 100
46  IF K > 39 THEN C0 =  SCRN( 0,1)

50  FOR M = K TO 0 STEP  - 1: REM  SPOSTAMENTI
60  COLOR=  SCRN( XA(M),YA(M))
70  PLOT XN(M),YN(M)
80  NEXT M

100  COLOR= C0: PLOT 0,0
110  IF K > 39 THEN 46: REM  RICIRCOLO
120  GOTO 42
```

Esercizi:

3.1 Programmare lo spostamento di un quadratino sulla diagonale.

3.2 Far rimbalzare un quadratino (che simula una palla) sui bordi dello schermo sonorizzando il rimbalzo.

3.3 Disegnare sullo schermo un motivo simmetrico rapportato all'asse verticale centrale e farlo salire o scendere.

Qualche comando particolare:

Cancellazione della pagina 1 in grafica bassa risoluzione

GR

Riempimento della pagina di testo 1 con nero su bianco

CALL - 1994 (le prime 20 linee)
CALL - 1998 (tutta la pagina di testo)

Cancellazione della attuale pagina grafica in alta risoluzione

CALL 62450 (indipendentemente dal modo)

Colorazione uniforme della pagina grafica in alta risoluzione nell'ultimo colore utilizzato (è necessario aver eseguito in precedenza un tracciamento con HPLOT 0,0):

CALL 62454

Modo alta risoluzione

Sono visualizzabili dei punti elementari in ragione di 280 posizioni orizzontali e 192 posizioni verticali nel modo non misto, oppure, nel modo misto, 280 x 160 con la possibilità di scrivere 4 linee di caratteri nella parte bassa dello schermo.

La colorazione dei punti risente di limitazioni tecniche (dovute alla scansione nei TV a colori) e a quelle imposte dal sistema Apple per la grafica a colori.

Si sa che un bit di memoria è associato a ciascun punto sullo schermo, e che in un byte sono raggruppati 7 punti consecutivi di una stessa linea e il bit più significativo di ciascun byte determina la classe del colore dei punti di quel byte.

Le due classi di colore sono così composte:

CC0 = { violetto, verde } (colori complementari)
CC1 = { blue, marrone }

Si distinguono, inoltre, in ciascun byte, due classi di punti:

CP = {b0, b2, b4, b6}

i bit dei punti situati nelle colonne pari:

CI = {b1, b3, b5}

i bit dei punti situati nelle colonne dispari dello schermo.

Nella classe dei colori CC0 = (violetto, verde):

- Il colore violetto può colorare solo i punti situati nelle colonne dispari, dunque i punti della classe CI.

- Il colore verde può colorare solo i punti situati nelle colonne pari, dunque i punti corrispondenti alla classe del tipo CP.

Se il bit che determina la classe del colore è a 1 allora è utilizzata la classe CC1 = (blue, marrone):

- Il colore blue è riservato ai punti delle colonne dispari (CI).

- Il colore marrone è riservato ai punti delle colonne pari (CP).

Il colore bianco è ottenuto sul televisore a colori dalla sovrapposizione di un punto CP (nero) e di un punto seguente CI (bianco).

Nel caso del TV in bianco e nero, programmando il tracciamento dei punti nel colore bianco (HCOLOR = 3), sono visualizzabili sullo schermo tutti i punti, pari o dispari qualunque sia la loro posizione orizzontale.

L'istruzione che definisce il colore di un punto o di un tracciamento in alta risoluzione si scrive:

HCOLOR = n° del codice del colore

Codice dei colori :	Colori possibili :	
Ø	NERO	
1	VIOLA	Colonne dispari
2	VERDE	Colonne pari
3	BIANCO	
4	NERO	
5	BLU	Colonne dispari
6	MARRONE	Colonne pari
7	BIANCO	

L'istruzione generale di tracciamento in alta risoluzione è:

HPLOT X,Y
per ottenere un solo punto

d'ascissa X (compresa tra 0 e 279) e
d'ordinata Y (compresa tra 0 e 191)

oppure

HPLOT X1,Y1 TO X2,Y2
per ottenere la retta con gli estremi (X1,Y1) e (X2,Y2)

oppure

HPLOT X1,Y1 TO X2,Y2 TO X3,Y3 TO X4,Y4 TO X1,Y1
per ottenere una linea spezzata chiusa

Esempi:

Tracciamento di un quadrato in alta risoluzione in pagina 2 in modo non misto

```
10  HGR2
20  HCOLOR= 3
30  HPLOT 0,0 TO 279,0 TO 279,191 TO 0,191 TO 0,0
40  END
```

Tracciamento di un reticolo di rette verticali

```
5  HGR2 : REM  CANCELLAZIONE E VISUALIZZIONE SULLA PAGINA 2
10  HCOLOR= 1: REM  VIOLA-DISPARI
20  FOR K = 0 TO 139
30  HPLOT 2 * K + 1,0 TO 2 * K + 1,20
40  NEXT K
50  HCOLOR= 2: REM  VERDE-PARI
60  FOR K = 0 TO 139
70  HPLOT 2 * K,30 TO 2 * K,50
80  NEXT K

100  HCOLOR= 3: REM  BIANCO-PARI E DISPARI
110  FOR K = 0 TO 139
120  HPLOT 2 * K,60 TO 2 * K,80
130  HPLOT 2 * K + 1,60 TO 2 * K + 1,80
140  NEXT K
```

Reticolo obliquo passante per il centro

```
10  HGR : REM  CANCELLAZIONE E VISUALIZZAZIONE SULLA PAGINA 1
20 P% = ( RND (1) * 5) + 2: REM  PASSO DA 2 A 6
30  FOR X = 0 TO 278 STEP P%
```

```
40   FOR K = 0 TO 1: REM  PARI E DISPARI
50   HCOLOR= 3 * K: REM  NERO - BIANCO
60   HPLOT X + K,0 TO 279 - X - K,159
70   NEXT K
80   NEXT X

100  FOR Y = 0 TO 158 STEP P%
110  FOR K = 0 TO 1: REM  COLONNE PARI E DISPARI
120  HCOLOR= 3 * K
130  HPLOT 279,Y + K TO 0,159 - Y - K
140  NEXT K
150  NEXT Y

200  FOR Z = 1 TO 500: NEXT Z
210  GOTO 10
```

Esercizi:

3.4 Disegnare un cubo in prospettiva

3.5 Scrivere un programma interattivo che, con l'aiuto di alcuni tasti (8 direzioni possibili), componga una figura con la lunghezza variabile dei segmenti (tasti numerici da 1 a 8).

GRAFICI (SHAPES)

Il sistema Apple consente di tenere in memoria un insieme di forme elementari (da 1 a 255), e di manipolare ognuna di queste forme (ingrandimento, inclinazione e posizioni programmate, quindi variabili) in una pagina grafica ad alta risoluzione.

Creazione e codifica di una figura

Ogni figura è composta da vettori limitati alle quattro direzioni (alto, basso,destra, sinistra). Ogni vettore può essere tracciato o meno, verrà simbolizzato il valore di un vettore con una delle quattro lettere H,B,D,G (maiuscole) se deve essere tracciato, o con una delle quattro lettere minuscole h,b,d,g (minuscole) se non deve essere tracciato.

Esempio: di un carattere alfabetico la lettera S ("·" rappresenta il punto più piccolo visibile sullo schermo)
("." rappresenta un vettore non tracciato : "X" l'origine del disegno).

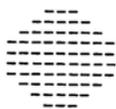
Codifica simbolica di questo disegno:
ddHGHBDBDDDHDHHGHGGHGHDH
DDDBHBddbbbbbbd

Questo tracciato è stato creato in scala unitaria: ogni vettore tracciato è un punto elementare di una linea TV e la sua direzione indica dove si troverà il punto seguente. Una volta registrata in memoria la figura, il sistema di tracciamento realizzerà automaticamente, su richiesta, un ingrandimento o un cambiamento di direzione.

Si vedrà una retta orientata a 45 gradi che diventa, secondo il valore dell'argomento nel fattore di scala, una bella....scala.

Esempio d'ingrandimento della lettera S con fattore 2:

I punti di una stessa linea orizzontale sono contigui, mentre i punti di una linea verticale sono separati da una interlinea.

Esempio di creazione di una figura piena:

La codifica necessita la scansione di tutti i punti. Sceglieremo un percorso circolare partendo dal basso verso il centro:

GHGHGHGHHH
DHDHDHDD
BDBDBDBBB
GBGBGBG
HHGHGHGH
DHDHD
BDBDBD
GBGG
HHGHDHBDBG

Per convertire le lettere utilizzate simbolicamente in informazioni binarie, si utilizza il seguente codice:

H 100	h 000	
B 110	b 010	
D 101	d 001	
G 111	g 011	

0 0	V 2	V 1
0 0	V 4	V 3

I codici dei vettori successivi di una figura, sono memorizzati in ragione di 2 vettori per byte, cominciando da destra: i due bit più a sinistra possono eventualmente contenere un vettore b, d oppure g. Un vettore h potrà essere registrato nel mezzo solo se i bit di sinistra sono significativi. Il primo vettore nullo segna la fine della figura. L'insieme delle figure costituisce il file delle figure che comprende una parte iniziale con dei valori caratteristici.

Registrazione di un file di figure

All'inizio del file vi è il numero delle figure definite (primo byte), un byte non utilizzato (secondo byte), un indice di due byte per determinare l'indirizzo d'inizio del codice relativo alla prima figura, un indice di due byte per determinare l'indirizzo d'inizio della seconda figura, ecc..., poi, dopo gli indici, i codici delle figure successive.

L'indice di riferimento alla codifica di una figura è relativo all'indirizzo d'inizio del file, sia S questo indirizzo; se l'indirizzo d'inizio della codifica della prima figura è A1, l'indice vale I1 = A1 - S.

Il primo dei due byte riservati ai valori di indice, corrisponde al valore meno significativo, il secondo al valore più significativo.

L'indirizzo di inizio di un file di figure deve necessariamente essere comunicato al sistema. Le memorie di indirizzo $E8 e $E9 sono riservate a questo scopo.

$E8 contiene i valori meno significativi dell'indirizzo d'inizio

$E9 contiene i valori più significativi dell'indirizzo d'inizio.

Il file dev'essere separato dalla zona di programma e dalla zona dati utilizzata. Iniziare il programma con l'istruzione HIMEM: l'indirizzo d'inizio del file, assicura che il programma non interferirà con la zona del file. L'esempio che segue introduce le figure in memoria leggendo le istruzioni DATA, elabora poi funzioni particolari mostrando l'effetto relativo al cambiamento di scala e alla direzione.

L'ingrandimento è ottenuto con l'istruzione SCALE = da 1 a 255. Il cambio di direzione è ottenuto con l'istruzione ROT = da 0 a 64 per passi variabili secondo la seguente tabella:

Valore per ROT	Orientamento
0	iniziale
16	90°
32	180°
48	270°

Maggiore è l'argomento della scala, più il passo relativo alla rotazione può essere piccolo.

Il tracciamento di una figura si ottiene con l'istruzione

DRAW I AT X,Y

I è il numero d'ordine della figura nel file.
X e Y sono le coordinate del punto di partenza del disegno.

Una istruzione permette di cancellare una figura particolare:

X DRAW I AT X,Y

Questa cancellazione è ottenuta con un tracciamento sovrapposto, con un colore complementare di ogni punto della figura iniziale.

```
5  HIMEM: 7676
10  REM  UTILIZZO DI UN FILE DI FIGURE
15  RESTORE
20 S = 7676: REM  INDIRIZZO D'IMPLEMENTAZIONE DEL FILE
30 PS =  INT (S / 256):MS = S - PS * 256
32  POKE 232,MS: POKE 233,PS: REM  $E8 E $E9 INIZIALIZZATI CON 7676
40  GOSUB 1000
45  FOR K = 1 TO N
46  GOSUB 50
47  NEXT K
48  GOTO 500
50  FOR I = 0 TO T(K) - 1
55  READ X
60  POKE S + I(K) + I,X
65  NEXT I
79  REM  LETTERA S
90  RETURN
500  HOME : HGR : HCOLOR= 3
510  ROT= 0
```

```
520  SCALE= 1: DRAW 1 AT 130,50: DRAW 1 AT 148,50
530  SCALE= 1: DRAW 2 AT 142,51
540 S = 0
543 A = 1
550  FOR X = 140 TO 0 STEP  - 10
555 S =  - 9 * X / 70 + 19
558 Y =  - X / 1.4 + 159
560  SCALE= S: DRAW 1 AT X,Y
562  IF Y <  = 120 THEN 565
563  SCALE= A: DRAW 2 AT 142,Y
564 A = A + 1
565  SCALE= S: DRAW 1 AT 175 - 1 / 4 * X,Y
570  NEXT X
580  FOR X = 0 TO 140 STEP 10
585 S =  - 9 * X / 70 + 19
588 Y =  - X / 1.4 + 159
590  SCALE= S: HCOLOR= 0: DRAW 1 AT X,Y
592  DRAW 1 AT 175 - 1 / 4 * X,Y
595  NEXT X
598  HCOLOR= 3: SCALE= 60
599  HPLOT 0,0: CALL 62454: HCOLOR= 0
600  FOR R = 0 TO 64
605  ROT= R
610  DRAW 3 AT 140,80
620  NEXT R
630  END
1000  READ N: REM  NUMERO DELLE FIGURE
1010  POKE S,N: POKE S + 1,0
1020  FOR F = 1 TO N
1030  READ T(F): REM  LUNGHEZZA DELLE FIGURE
1040  NEXT F
1100 I(1) = 2 + 2 * N:J(1) = 0
1110 I(2) = T(1) + I(1):J(2) = 0
1130 I(3) = T(2) + I(2):J(3) = 0
1150  FOR F = 1 TO N
1160  POKE S + 2 * F,I(F)
1165  POKE S + 2 * F + 1,J(F)
1170  NEXT F
1200  RETURN
1300  DATA  3,20,36,3
1310  DATA  9,60,52,53,45,44,36,39,63,60,44,44,45,38,14,17,18,18,10,0
1320  DATA  39,39,39,39,36,37,37,37,45,46,46,46,54,62,62,62,62,32,39
1330  DATA  39,39,37,37,37,46,46,46,62,62,62,36,39,37,46,62,00
1350  DATA  44,62,00
```

Memorizzazione di una figura

Su cassetta

E' necessario trasferire su cassetta la zona di memoria dov'è memorizzata la figura. Questa figura ha:

- un indirizzo d'inizio   S
- un indirizzo finale   F
- una lunghezza   F - S

La differenza F - S dev'essere registrata nell'indirizzo $00 e $01, poi la registrazione si "lancia", da Monitor:

*0.1W S.FW.

Esempio:

S = $1DFC
F = $1E1A
F - S = $1E
*00 : 1E00
*0.1W 1DFC.1E1AW

Su dischetto

Il trasferimento di una zona di memoria si effettua con il comando

BSAVE FF, Aa,L1

a è l'indirizzo d'inizio della zona,
1 è la sua lunghezza.

Esempio:

a = $1DFC
1 = $1F
] BSAVE FF, A$1DFC,L$1F

registrazione su dischetto del file di figura chiamato FF.

CARICAMENTO IN MEMORIA DI UN FILE DI FIGURA

Da cassetta

Il file su cassetta è automaticamente posizionato sotto la parte più alta della memoria RAM disponibile (sotto HIMEM), con l'istruzione: SHLOAD.

Questa istruzione assicura il trasferimento dall'indirizzo d'inizio del file ai sottoprogrammi di tracciamento delle figure.

Dopo il caricamento, il valore di HIMEM viene corretto automaticamente per proteggere il file, cioè HIMEM è uguale all'indirizzo d'inizio del file.

Da dischetto

Il comando DOS per il caricamento di un file binario in memoria RAM è:

] BLOAD FF I,Aa,L1

I parametri a e 1 sono opzionali per consentire il cambiamento dell'indirizzo d'implementazione in memoria. Se questi parametri non sono menzionati, il file è posizionato all'indirizzo specificato nel comando BSAVE FF.

APPENDICE I

PRONTUARIO APPLESOFT

Matrici e stringhe

DIM A(X,Y,Z)	*Specifica i valori massimi degli indici della matrice A e riserva una zona memoria di (X+1)*(Y+1)*(Z+1) elementi reali, cominciando con A(0,0,0).*
DIM A$(X,Y)	*Specifica i valori massimi degli indici di A$ che possono contenente (X+1)*(Y+1) stringhe ciascuna composta con un massimo di 255 caratteri.*
LEN (A$)	*Restituisce il numero di caratteri di cui è composta A$.*
STR$ (X)	*Restituisce il valore numerico di X convertito in stringa.*
VAL (A$)	*Restituisce A$ fino al primo carattere non numerico, in valore numerico.*
CHR$ (X)	*Restituisce il carattere il cui codice ASCII vale X.*
ASC (A$)	*Restituisce il codice ASCII del primo carattere di A$.*
LEFT$ (A$,X)	*Restituisce gli X caratteri più a sinistra di A$.*
RIGHT$ (A$,X)	*Restituisce gli X caratteri più a destra di A$.*
MID$ (A$,X,Y)	*Restituisce gli Y caratteri di A$ partendo dal Xiesimo carttere.*
+	*Operatore di concatenazione di strighe di caratteri.*
STORE A	*Registra una matrice di dati su cassetta.*
RECALL B	*Carica in memoria una matrice registrata su cassetta. La matrice B dev'essere correttamente dimensionata.*

Comandi di I/O

INPUT A$	*Visualizza un punto di domanda sullo schermo e attende che una stringa di caratteri sia impostata da tastiera.*
INPUT "XYZ";A	*Visualizza XYZ e attende che un valore numerico sia impostato da tastiera.*
GET A$	*Attende un solo carattere da tastiera senza RETURN per l'assegnazione ad A$.*
DATA X,"Y",Z	*Stabilisce una lista di dati da leggere con l'istruzione READ.*
READ A$	*Assegna ad A$ dato successivo della lista DATA.*
RESTORE	*Riprende la lettura dall'inizio della lista DATA.*
PRINT "X = "X;	*Visualizza la stringa X = seguita dal valore della variabile X. Il punto e virgola concatena gli elementi. La virgola separa la linea in tre campi. Il simbolo ? può sostituire l'istruzione PRINT.*
IN # 3	*Indica i dati che devono essere inviati alla periferica collegata allo slot numero 3.*
PR # 6	*Cerca i dati che devono essere inviati alla periferica collegata allo slot numero 6.*

Comandi di salto

GO TO 347	*Salta alla linea 347.*
IF X=3 THEN STOP	*Se l'affermazione X=3 è vera (non nulla) tutti gli ordini che seguono THEN vengono eseguiti. Se l'affermazione è falsa, l'esecuzione salta alla linea seguente.*

FOR X = 1 TO 20 STEP 4...NEXT X

Esegue tutte le linee d'istruzione comprese tra FOR e NEXT, prima con X=1 poi con X=5, X=9 ecc...fino a X < 20. Quindi l'esecuzione proseguirà con la linea che segue l'istruzione NEXT. Se non specificato STEP vale 1.

NEXT X	*Definisce il "fondo" del ciclo FOR...NEXT. X è opzionale.*

GO SUB 33 — *Salta al sotto-programma della linea 33.*

RETURN — *Segna la fine di un sotto-programma e rinvia l'esecuzione all'istruzione successiva che segue GO SUB.*

POP — *Ignora un indirizzo dalla catasta dei RETURN.*

ON X GOTO 38,12,45 — *Salta al Xiesima linea del programma. Se X = 2 salta alla linea 12.*

ON X GOSUB 38,12,45 — *Salta al sotto-programma della Xiesima linea del programma.*

ON ERR GOTO 4500 — *Gli errori successivi a questa istruzione provocano un salto alla linea 4500 in modo da visualizzare un messaggio di errore e arrestare l'esecuzione del programma.*

RESUME — *Inserito nel sotto-programma di trattamento dell'errore provoca il ritorno all'istruzione che ha dato luogo all'errore.*

Grafici

Bassa risoluzione

GR — *Attiva il modo grafico a bassa risoluzione. Cancella la pagina grafica (40x40). Lascia 4 linee per i caratteri nella parte bassa dello schermo.*

COLOR = X — *Fissa il colore (valore da 0 a 15).*

PLOT X,Y
$0 \leqslant X \leqslant 39$
$0 \leqslant Y \leqslant 39$ — *Pone un quadratino colorato nella posizione X dal bordo sinistro dello schermo e nella posizione Y dalla parte alta dell'area grafica.*

HLIN X1,X2 AT Y — *Traccia una linea orizzontale da (X1,Y) a (X2,Y).*

VLIN Y1,Y2 AT X — *Traccia una linea verticale da (X,Y1) a (X,Y2).*

SCRN (X,Y) — *Restituisce il colore in X,Y.*

HGR	*Attiva il modo grafico ad alta risoluzione. Pagina 1 (280 x 160).*
HGR2	*Pagina 2 (280 x 192) e cancella la pagina corrispondente.*
HCOLOR = X	*Fissa il colore del successivo punto da tracciare.*
HPLOT X,Y $0 \leqslant X \leqslant 280$ $0 \leqslant Y \leqslant 192$	*Pone un punto colorato nella posizione X dal bordo sinistro dello schermo e nella posizione Y dalla parte alta dell'area grafica.*
HPLO + T X1,Y1 TO X2,Y2	*Traccia una linea retta tra X1,Y1 e X2,Y2. Questo comando può essere esteso ad altri punti....TO X3,Y3.*
SHLOAD	*Carica in memoria una figura (shape) da cassetta.*
DRAW 3 AT X,Y	*Traccia la figura numero 3 della matrice precedentemente caricata al punto X,Y nel colore fissato con HCOLOR.*
XDRAW 3 AT X,Y	*Traccia la figura numero 3 al punto X,Y, il colore di ogni punto è il colore complementare del punto sullo schermo.*
ROT = X	*Fissa la direzione della figura* *ROT = 0 → verticale* *ROT = 16 → 90°; ROT = 32 → 180°*
SCALE = X	*Fissa il fattore di scala della figura (da 1 a 255).*

Le paddle e gli effetti sonori

PDL (X)	*Restituisce la posizione, da 0 a 255, della paddle X (0 o 1).*
PEEK (X — 16287)	*Se > 127, il pulsante X (della paddle 0 o 1) è stato premuto.*
PEEK (— 16336)	*Emette un "clik" dall'altoparlante.*

Funzioni matematiche

SIN (X)	*Restituisce il seno di X espresso in radianti.*

COS (X)	*Restituisce il coseno di X espresso in radianti.*
TAN (X)	*Restituisce la tangente di X espressa in radianti.*
ATN (X)	*Restituisce l'arcotangente di X espresso in radianti.*
INT (X)	*Restituisce la parte intera di X.*
RND (1)	*Restituisce un numero reale casuale compreso tra 0 e 0.999 999 999.*
RND (0)	*Restituisce l'ultimo numero casuale generato dall'istruzione RND (1).*
RND (— 3)	*Fornisce 4.482171 E-08. Vi è un numero diverso per ogni argomento negativo. Successivamente, RND (1) produrrà sempre la stessa sequenza.*
SGN (X)	*Fornisce - X –1 per X* < 0; 0 se X » 0; 1 se X > *0.*
ABS (X)	*Restituisce il valore assoluto di X.*
SQR (X)	*Restituisce la radice quadrata di X.*
EXP (X)	*Restituisce e (2.718289) alla potenza di X.*
LOG (X)	*Restituisce il logaritmo naturale di X.*

Variabili semplici

Tipo	Nome	Estremi
Reali	AB	+/—9.99999999 E +37
Interi	AB%	+/—32767
Stringhe	AB$	da 0 a 255 caratteri

A è una lettera, B è una lettera o un numero. Un nome di variabile si può scrivere con più di due caratteri, ma sono considerati solo i primi due: AB% e AB3QS% sono considerate identiche.

Matrici di variabili

Tipo	Nome di un elemento
Reale	AB (3, 12, 7)
Intero	AB% (3, 12, 7)
Stringa	AB$ (3, 12, 7)

La dimensione di una matrice è limitata solo allo spazio disponibile in memoria.

Operatori algebrici

=	*Assegnazione di un valore a una variabile*
—	*Negazione*
^	*Elevazione a potenza*
*	*Moltiplicazione*
/	*Divisione*
+	*Addizione*
—	*Sottrazione*

Operatori logici e di comparazione

=	*Uguaglianza*
<>	*diverso da*
<	*Minore di*
>	*Maggiore di*
<=	*Minore o uguale a*
>=	*Maggiore o uguale a*

Comandi

LOAD	*Carica da cassetta un programma in memoria.*
SAVE	*Registra il programma in memoria su cassetta.*
NEW	*Cancella il programma in memoria e tutte le variabili.*
RUN	*Esegue il programma in memoria partendo dalla linea con il numero inferiore. Cancella tutte le variabili.*
RUN 477	*Esegue il programma partendo dalla linea 477.*
STOP	*Arresta il programma e visualizza il numero di linea corrispondente.*
END	*Arresto del programma senza visualizzazione di messaggi.*
Ctrl - C	*Arresto immediato di un programma o di un listing.*
RESET	*Arresto di un programma con il rinvio del controllo al BASIC.*
CONT	*Riprende l'esecuzione di un programma interrotto con STOP, END o Ctrl-C.*
PEEK (A)	*Restituisce il contenuto della locazione di memoria di indirizzo A (espressa in decimale).*
POKE A,13	*Modifica il contenuto della locazione di memoria di indirizzo A con il valore 13.*
CALL X	*Salta ad un sotto-programma scritto in linguaggio macchina che inizia all'indirizzo X.*
WAIT X,Y,Z	*Attende che ((X) ⊕ (Z)).(Y) sia diverso da 0.* *⊕ OR esclusivo* *. e logico* *(X) contenuto della memoria X.*
USR (X)	*Passa il valore di X ad un sotto-programma in linguaggio macchina.*
HIMEM:	*Specifica l'indirizzo più alto di memoria disponibile.*
LOMEM:	*Specifica l'indirizzo più basso disponibile per i dati del programma.*

Stampa e tabulazione

LIST	*Visualizza tutte le linee d'istruzione del programma.*
LIST X — Y	*Visualizza la parte di programma compresa tra X e Y.*
DEL X,Y	*Cancella tutte le linee comprese tra X e Y.*

Editing

REM XYZ	*Consente la scrittura di commenti che sono ignorati durante l'esecuzione del programma.*
VTAB Y	*Posiziona il cursore sulla linea Y (da 1 a 24).*
HTAB X	*Posiziona il cursore sulla colonna X (da 1 a 40).*
TAB (X)	*In una istruzione PRINT, posiziona il cursore sulla colonna X (da 1 a 40).*
POS (0)	*Restituisce la posizione attuale del cursore (da 0 a 39).*
SPC (X)	*In una istruzione PRINT, pone X spazi tra due entità da visualizzare.*
HOME	*Cancella il contenuto dello schermo e posiziona il cursore in alto a sinistra.*
CLEAR	*Setta tutte le variabili a 0 e tutte le stringhe di caratteri vuote.*
FRE (0)	*Rinvia il numero di byte ancora disponibile.* *Cancella le stringhe di caratteri abbandonate.*
FLASH	*Attiva la visualizzazione lampeggiante.*
INVERSE	*Attiva la visualizzazione in campo bianco e caratteri neri.*
NORMAL	*Annulla i modi FLASH e INVERSE.*
SPEED= X	*Regola la velocità di visualizzazione dei caratteri da 1 a 255.*
esc K	*Sposta il cursore di una posizione verso destra.*

esc J	*Sposta il cursore di una posizione verso sinistra.*
esc M	*Sposta il cursore di una posizione verso il basso.*
esc I	*Sposta il cursore di una posizione verso l'alto.*
→	*Memorizza il carattere sotto il cursore e lo sposta di una posizione verso destra.*
←	*Annulla il carattere precedentemente impostato e sposta il cursore di una posizione verso sinistra.*
Ctrl - X	*Annulla la linea in corso d'impostazione.*
POKE 33,33	*Riduce lo schermo a 33 caratteri per linea senza l'inserimento di spazi supplementari nei listing. Impostare TEXT per annullare il comando.*

Funzione predefinite

DEF FN A(X) = X + 23 / X
Definisce una funzione FNA. Con la seguente definizione il valore di X sarà sostituito nell'espressione FNA (4) restituendo 9.75.

APPENDICE II

LE PAROLE RISERVATE ALLE ISTRUZIONI APPLESOFT E LORO CODIFICA INTERNA

Decimale	Esadecimale	Parola riservata
128	80	END
129	81	FOR
130	82	NEXT
131	83	DATA
132	84	INPUT
133	85	DEL
134	86	DIM
135	87	READ
136	88	GR
137	89	TEXT
138	8A	PR#
139	8B	IN#
140	8C	CALL
141	8D	PLOT
142	8E	HLIN
143	8F	VLIN
144	90	HGR2
145	91	HGR
146	92	HCOLOR=
147	93	HPLOT
148	94	DRAW
149	95	XDRAW
150	96	HTAB
151	97	HOME
152	98	ROT=
153	99	SCALE=
154	9A	SHLOAD
155	9B	TRACE
156	9C	NOTRACE
157	9D	NORMAL
158	9E	INVERSE
159	9F	FLASH
160	A0	COLOR=
161	A1	POP
162	A2	VTAB
163	A3	HIMEM:
164	A4	LOMEM:
165	A5	ONERR
166	A6	RESUME
167	A7	RECALL
168	A8	STORE
169	A9	SPEED=
170	AA	LET
171	AB	GOTO
172	AC	RUN
173	AD	IF
174	AE	RESTORE
175	AF	&
176	B0	GOSUB
177	B1	RETURN
178	B2	REM
179	B3	STOP
180	B4	ON
181	B5	WAIT

Decimale	Esadecimale	Parola riservata
182	B6	LOAD
183	B7	SAVE
184	B8	DEF
185	B9	POKE
186	BA	PRINT
187	BB	CONT
188	BC	LIST
189	BD	CLEAR
190	BE	GET
191	BF	NEW
192	C0	TAB(
193	C1	TO
194	C2	FN
195	C3	SPC(
196	C4	THEN
197	C5	AT
198	C6	NOT
199	C7	STEP
200	C8	+
201	C9	-
202	CA	*
203	CB	/
204	CC	^
205	CD	AND
206	CE	OR
207	CF	>
208	D0	=
209	D1	<
210	D2	SGN
211	D3	INT
212	D4	ABS
213	D5	USR
214	D6	FRE
215	D7	SCRN(
216	D8	PDL
217	D9	POS
218	DA	SQR
219	DB	RND
220	DC	LOG
221	DD	EXP
222	DE	COS
223	DF	SIN
224	E0	TAN
225	E1	ATN
226	E2	PEEK
227	E3	LEN
228	E4	STRØ
229	E5	VAL
230	E6	ASC
231	E7	CHRØ
232	E8	LEFTØ
233	E9	RIGHTØ
234	EA	MIDØ

APPENDICE III

CODICE ASCII DEI CARATTERI

CAR.	Codifica ASCII	
	Decimale	Esadecimale
1	49	31
2	50	32
3	51	33
4	52	34
5	53	35
6	54	36
7	55	37
8	56	38
9	57	39
ø	48	30
:	58	3A
-	45	2D
;	59	3B
,	44	2C
.	46	2E
/	47	2F

CAR.	Codifica ASCII	
	Decimale	Esadecimale
!	33	21
"	34	22
#	35	23
$	36	24
%	37	25
&	38	26
'	39	27
(	40	28
)	41	29
Spazio	32	20
*	42	2A
=	61	3D
+	43	2B
<	60	3C
>	62	3E
?	63	3F

Queste tabelle forniscono i valori ASCII dei caratteri generati dalla tastiera e memorizzati con INPUT C$ o GET C$.

* Alla visualizzazione sullo schermo, questi codici restano evidentemente validi, ma lo stesso carattere può essere visualizzato con il codice della tabella di cui sopra, modulo $(64)_{10}$ o $(40)_{16}$ cioè:

PRINT CHR$(48) o PRINT CHR$(48+64) o PRINT CHR$(48+128) o PRINT CHR$(48+192) visualizzerà sempre 0.

Codici ASCII dei caratteri alfabetici

CAR.	Codifica ASCII	
	Decimale	Esadecimale
@	64	40
A	65	41
B	66	42
C	67	43
D	68	44
E	69	45
F	70	46
G	71	47
H	72	48
I	73	49
J	74	4A
K	75	4B
L	76	4C

CAR.	Codifica ASCII	
	Decimale	Esadecimale
M	77	4D
N	78	4E
O	79	4F
P	80	50
Q	81	51
R	82	52
S	83	53
T	84	54
U	85	55
V	86	56
W	87	57
X	88	58
Y	89	59
Z	90	5A

SHIFT P	64	40	= @
SHIFT N	94	5E	= ^
SHIFT M	93	5D	=]

Ogni altra lettera unita al tasto SHIFT non modifica il carattere o il codice.

* Aggiungendo $(128)_{10}$ o $(80)_{16}$ a ciascuno di questi codici, si ritrovano gli stessi caratteri visualizzati sullo schermo.

Esempio PRINT CHR$(65), CHR$(193) fornirà sempre A.

Caratteri di controllo
Premere CTRL e i tasti

	Codifica ASCII	
	Decimale	Esad.
ctrl @	0	00
ctrl A	1	01
ctrl B	2	02
ctrl C	3	03
ctrl D	4	04
ctrl E	5	05
ctrl F	6	06
ctrl G	7	07
ctrl H	8	08
ctrl I	9	09
ctrl J	10	0A
ctrl K	11	0B
ctrl L	12	0C
ctrl M	13	0D
ctrl N	14	0E
ctrl O	15	0F
ctrl P	16	10
ctrl Q	17	11
ctrl R	18	12
ctrl S	19	13
ctrl T	20	14
ctrl U	21	15
ctrl V	22	16
ctrl W	23	17
ctrl X	24	18
ctrl Y	25	19
ctrl Z	26	1A
ctrl SHIFT M	29	1D
ctrl ^	30	1E

Altri tasti della tastiera

RETURN	13	OD	= ctrl M
ESC	27	1B	
⟵	8	08	= ctrl H
⟶	21	15	= ctrl U

* Questo insieme di caratteri non è visualizzabile sullo schermo.

Caratteri non disponibili sulla tastiera ma utilizzabili in memoria per la visualizzazione sullo schermo

Codifica ASCII		Carattere visualizzato
Decimale	Esadecimale	
92	5C	\
91	5B	[
95	5F	(sottolineatura)

Esempio di codifica su stampante
[OKI Microline 80] (80 car./linea)

DEC		DEC		DEC		DEC		DEC	
32		51	3	7Ø	F	89	Y	1Ø8	l
33	!	52	4	71	G	9Ø	Z	1Ø9	m
34	"	53	5	72	H	91	↑	11Ø	n
35		54	6	73	I	92	↓	111	o
36	$	55	7	74	J	93	←	112	p
37	%	56	8	75	K	94	→	113	q
38	&	57	9	76	L	95	_	114	r
39	'	58	:	77	M	96		115	s
4Ø	(	59	;	78	N	97	a	116	t
41	)	6Ø	<	79	O	98	b	117	u
42	*	61	=	8Ø	P	99	c	118	v
43	+	62	>	81	Q	1ØØ	d	119	w
44	,	63	?	82	R	1Ø1	e	12Ø	x
45	-	64		83	S	1Ø2	f	121	y
46	.	65	A	84	T	1Ø3	g	122	z
47	/	66	B	85	U	1Ø4	h	123	{
48	Ø	67	C	86	V	1Ø5	i	124	|
49	1	68	D	87	W	1Ø6	j	125	}
5Ø	2	69	E	88	X	1Ø7	k	126	-
								127	

ATTENZIONE:

A partire dal codice 91, i caratteri stampati sono differenti da quelli visualizzati sullo schermo.

APPENDICE IV

RICHIAMO SULLE BASI DI NUMERAZIONE BINARIA ED ESADECIMALE

Un numero intero N, espresso nel sistema di numerazione di base B ha per valore decimale:

$$N = b_{n-1}*B^{n-1} + b_{n-2}*B^{n-2} + ... + b_1*B^1 + b_0*B^0$$

con le seguenti notazioni:

*	=	Moltiplicazione.
B^n	=	B elevato alla potenza di n; cioè B*B*B*...*B n volte; $B^0 = 1$.
b_n		Cifra situata alla n+1iesima posizione da destra.
N_B	=	Rappresentazione del valore N in base B.
N_B	=	$b_{n-1}\ b_{n-2} ... b_1\ b_0$

Esempi:

$B = 10,\ N_{10} = 1024$, $N = 1*10^3 + 0*10^2 + 2*10^1 + 4*10^0$
$B = 2,\ N_2 = 10000000000$, $N = 1*2^{10}$
$B = 16,\ N_{16} = 400$, $N = 4*16^2 + 0*16^1 + 0*16^0$

Le cifre bi appartengono necessariamente all'insieme di cifre caratterizzate dalla base B. Sono tante quante indicate da B.

$B = 10\ \ b_i \in \{0,1,2,3,4,5,6,7,8,9\}$
$B = 2\ \ b_i \in \{0,1\}$
$B = 16\ \ b_i \in \{0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F\}$

Valori decimali di cifre esadecimali (B=16):

A = 10; B = 11; C = 12; D = 13; E = 14; F = 15

Valori binari delle cifre esadecimali:

```
0 = 0000     4 = 0100     8 = 1000     C = 1100
1 = 0001     5 = 0101     9 = 1001     D = 1101
2 = 0010     6 = 0110     A = 1010     E = 1110
3 = 0011     7 = 0111     B = 1011     F = 1111
```

Modifica della base di numerazione

Conversione decimale/binaria

Dividere successivamente N per 2. I resti delle divisioni intere successive sono i valori binari cercati, dai meno significativi ai più significativi.

37/2 = 18 resto 1
18/2 = 9 resto 0
9/2 = 4 resto 1 $\quad 37_{10} = 1\ 0\ 0\ 1\ 0\ 1_2$
4/2 = 2 resto 0
2/2 = 1 resto 0
1/2 = 0 resto 1

Conversione binaria/esadecimale

Raggruppare per quattro iniziando dalle cifre binarie di destra. Ogni gruppo di quattro è una cifra esadecimale.

$1\ 0\ 0\ 1\ 0\ 1_2 = 1\ 0\quad 0\ 1\ 0\ 1_2 = 25_{16}$

Conversione esadecimale/binaria

Far corrispondere ad ogni cifra esadecimale la combinazione di quattro cifre equivalenti.

$1F_{16} = 0\ 0\ 0\ 1 \quad 1\ 1\ 1\ 1_2$

VALORI ESADECIMALI MOLTO CARATTERISTICI E LORO EQUIVALENTI

Ø	Ø		Ø		
F	1111		15		
FF	11111111		255	$= 2^8-1$	
1ØØ	1ØØØØØØØØ		256	$= 16^2$	
4ØØ			1 024	$= 2^{10}$ =	1 K
8ØØ			2 048	$= 2^{11}$ =	2 K
1ØØØ			4 096	$= 2^{12}$ =	4 K
2ØØØ			8 192	$= 2^{13}$ =	8 K
4ØØØ			16 384	$= 2^{14}$ =	16 K
8ØØØ			32 768	$= 2^{15}$ =	32 K
CØØØ			49 152		= 48 K
FØØØ		$= 15*16^3 = 15\ *$	4 096		= 60 K
FFFF		$= 16^4-1 =$	65 535		= 64 K

APPENDICE V

CODICI D'ERRORE E MESSAGGI

Codice	messaggio	spiegazione
0	NEXT WITHOUT FOR	*La variabile dell'istruzione NEXT non corrisponde a quella specificata in FOR.*
16	SYNTAX ERROR	*Mancanza di parentesi in una espressione oppure la punteggiatura non è corretta.*
22	RETURN WITHOUT GOSUB	*Manca un GOSUB.*
42	OUT OF DATA	*Mancano dei dati nella lista DATA perchè l'istruzione READ possa essere eseguita.*
53	ILLEGAL QUANTITY	*Il parametro di una funzione eccede il suo limite. Esempio indice negativo per una variabile indicizzata.*
69	MEMORY OVERFLOW	*Il risultato di un calcolo è troppo elevato per poter essere rappresentato in BASIC.*
77	OUT OF MEMORY	*La capacità di memoria non è sufficiente: il programma è troppo grande o le variabili troppo numerose o LOMEM troppo alto oppure HIMEM troppo basso.*
90	UNDEF'D STATEMENT	*Il numero di linea con un'istruzione GOTO, GOSUB oppure THEN non esiste.*
107	BAD SUBCRIPT	*Il riferimento di un elemento di una matrice è fuori dimensione.*
120	REDIM'D ARRAY	*Ci sono due istruzioni di dimensionamento della stessa matrice.*
133	DIVISION BY ZERO	*E' un errore voler dividere per zero.*

163	TYPE MISMATCH	*Non è possibile assegnare un valore numerico ad una stringa di caratteri, oppure un valore alfanumerico ad un numero.*
176	STRING TOO LONG	*Una stringa non può contenere più di 255 caratteri.*
191	FORMULA TOO COMPLEX	*Non è possibile domandare l'esecuzione di oltre due istruzioni nella forma IF "XX" THEN.*
224	UNDEF'D FUNCTION	*Una funzione è utilizzata nel programma senza essere stata precedentemente definita.*
254	REENTER	*Errata risposta ad una istruzione INPUT.*
255	BREAK	*E' stata eseguita un'interruzione del programma con CTRL-C.*

APPENDICE VI

CORREZIONE DEGLI ESERCIZI

2.1 Codifica interna dei numeri interi

```
]10 A%=4096
]RUN
]CALL -151
*800.817
0800 - 00/0D 08 0A 00 41 25 D0 //codifica dell'istruzione 10
0808 - 34 30 39 36 00 00/00 0A
0810 - C1 80 [10 00] 00 00 00 49
*812 : F0 00           (modifica del valore di A)
*'Ctrl C'
]PRINT A%
-4096
```

2.2 Conversione di 2 elevato alla 15 in codice interno (esponente, mantissa)

La formula $N=(M+0.5)2^{E-128}$ permette di trovare M = 0 e
$2^{15} = 2^{-1}.2^{E-128}$ porta ad un esponente E=15+1+128 =144.

Inoltre $(144)_{10} = (90)_{16}$

Dunque $2^{15} \rightarrow$ | 90 | 00 | 00 | 00 | 00 |

2.3 Traduzione di 00 FF FF FF FF in notazione virgola flottante

Questo valore è compreso nell'intervallo di numeri troppo piccoli in valore assoluto e sarà visualizzato come - 0.

2.4 Generazione automatica di operazioni particolari

```
5 X = 12345678:X$ = "(12345678+1)X"
10  FOR I = 1 TO 9
20  PRINT X$;9 * I;"=";(X + 1) * 9 * I
30  NEXT I

]RUN
(12345678+1)X9=111111111
(12345678+1)X18=222222222
(12345678+1)X27=333333333
(12345678+1)X36=444444444
(12345678+1)X45=555555555
(12345678+1)X54=666666666
(12345678+1)X63=777777777
(12345678+1)X72=888888888
(12345678+1)X81=999999999
```

2.5 Vocaboli

```
10  REM  ESERCIZIO 2.5
11  REM  PALINDROME
15 :
20  INPUT A$
25 L =  LEN (A$):B$ = ""
26  IF L THEN 30
27  END
30  FOR I = L TO 1 STEP  - 1
40 B$ = B$ +  MID$ (A$,I,1)
50  NEXT I
55 :
60  IF B$ = A$ THEN  PRINT A$;" E' PALINDROMA"
70  GOTO 10
80 :
100  REM  PALINDROME ARTIFICIALI
101 :
105  FOR L = 2 TO 10 STEP 2
108 A$ = ""
110  FOR K = 1 TO L / 2
113 B$ =  CHR$ (65 + 26 *  RND (1))
115 A$ = B$ + A$ + B$
120  NEXT K
130  PRINT A$
150  NEXT L
160  GOTO 100

]RUN
?ADA
ADA E' PALINDROMA
?APPLE
?
```

2.6 Anagrammi

```
10  REM  ANAGRAMMI
12  INPUT A$
15  FOR K1 = 1 TO 4
17  FOR K2 = 1 TO 4
18  IF K2 = K1 THEN 33
20  FOR K3 = 1 TO 4
21  IF (K3 = K1) OR (K3 = K2) THEN 32
22  FOR K4 = 1 TO 4
23  IF (K4 = K1) OR (K4 = K2) OR (K4 = K3) THEN 31
24 :
30  PRINT  MID$ (A$,K1,1); MID$ (A$,K2,1); MID$ (A$,K3,1); MID$ (A$,K4,1);
    "  ";
31  NEXT K4
32  NEXT K3
33  NEXT K2
34  PRINT
35  NEXT K1
```

```
]RUN
?LUNA
LUNA   LUAN   LNUA   LNAU   LAUN   LANU
ULNA   ULAN   UNLA   UNAL   UALN   UANL
NLUA   NLAU   NULA   NUAL   NALU   NAUL
ALUN   ALNU   AULN   AUNL   ANLU   ANUL
```

2.7 Lunghezza di una stringa

```
10  REM  LUNGHEZZA
20  INPUT A$:X$ = A$: GOSUB 100:A$ = X$
30  INPUT B$:X$ = B$: GOSUB 100:B$ = X$
40  INPUT C$:X$ = C$: GOSUB 100:C$ = X$
50  PRINT A$;B$;C$
55  END
100 X$ = " " + X$
110  FOR K =  LEN (X$) + 1 TO 15:X$ = X$ + " ": NEXT K
120  RETURN

]RUN
?LARGHEZZA
?LUNGHEZZA
?ALTEZZA
 LARGHEZZA       LUNGHEZZA       ALTEZZA
```

2.8 Visualizzazione di numeri reali in virgola fissa

```
10  REM  ESERCIZIO 2.8
12  REM  VISUALIZZAZIONE DI NUMERI REALI IN VIRGOLA FLOTTANTE
13  REM  SENZA L'ELIMINAZIONE DEGLI ZERI NON SIGNIFICATIVI DELLA PARTE DEC
    IMALE
14 :
20  INPUT N:X$ =  STR$ (N): GOSUB 100
22 N =  VAL (X$)
25  REM  INCOLONNAMENTO A DESTRA SULLA COLONNA 15
30  PRINT ; SPC( 15 -  LEN (X$));X$
40  PRINT : GOTO 20
50 :
100  REM  RICERCA DEL PUNTO DECIMALE
105  FOR K = 1 TO  LEN (X$)
110  IF  MID$ (X$,K,1) = "." THEN 150
120  NEXT K
130  GOTO 200: REM  NUMERO INTERO
140 :
150  IF  MID$ (X$, LEN (X$) - 1,1) = "." THEN 300
160  IF  MID$ (X$, LEN (X$) - 2,1) = "." THEN 1000
162  REM  PIU' DI DUE CIFRE DECIMALI
```

```
163  REM  ARROTONDARE AL CENTESIMO
165 N =  INT (N * 100 + 0.5) / 100
170 X$ =  STR$ (N)
180  GOTO 100
200 X$ = X$ + ".00": GOTO 1000
300 X$ = X$ + "0"
1000  RETURN

]RUN
?3.14159
           3.14

?100.5
         100.50

?9999999.99
    10000000.00

?0.275
            .28

?

BREAK IN 20
```

2.9 Medie

```
 10   REM  RISULTATI D'ESAME
 20   REM  ACQUISIZIONE DEI DATI
 25   DIM T(6,10),M(10)
 28   FOR M = 1 TO 6
 30   FOR E = 1 TO 10
 35   READ T(M,E)
 40   NEXT E,M
 45   REM  MATERIA 1
 46   DATA  5,6,7,8,12,13,11,10,9,8
 47   REM  MATERIA 2
 48   DATA   13,11.5,12,14.5,11,7,13,8.5,9.5,10
 49   REM  MATERIA 3
 50   DATA  2.5,0,3,9,4,12,11.5,15,12,14
 51   REM  MATERIA 4
 52   DATA  8,4.5,10,13,16,17.5,12,6,7,7
 53   REM  MATERIA 5
 54   DATA  9,10,13,16,19,2,5.5,7.5,9,10
 55   REM  MATERIA 6
 56   DATA  12,15,11,3.5,7,9,10,11,13,17
 60 :
 62   REM  ELABORAZIONE DEI DATI
```

```
65  FOR E = 1 TO 10
66 M(E) = 0
70  FOR M = 1 TO 6
72 M(E) = M(E) + T(M,E)
74  NEXT M
75 M(E) = M(E) / 6
77  NEXT E
80 :
82  REM  RISULTATI
85  FOR E = 1 TO 10
90  PRINT ; SPC( 2 -  LEN ( STR$ (E)));E;
100  FOR M = 1 TO 6
110 X$ =  STR$ (T(M,E))
112 N = T(M,E): GOSUB 1000
114  PRINT "!"; SPC( 4 -  LEN (X$));X$;
116  NEXT M
122 N =  INT (M(E) * 2 + .5) / 2
123 X$ =  STR$ (N): GOSUB 1000
124  PRINT " !"; SPC( 5 -  LEN (X$));X$
130  NEXT E
150  END
999  REM  MESSA A PUNTO...
1000  IF N =  INT (N) THEN X$ = X$ + ".0"
1010  RETURN

]RUN
 1! 5.0!13.0! 2.5! 8.0! 9.0!12.0 !  8.5
 2! 6.0!11.5! 0.0! 4.5!10.0!15.0 !  8.0
 3! 7.0!12.0! 3.0!10.0!13.0!11.0 !  9.5
 4! 8.0!14.5! 9.0!13.0!16.0! 3.5 ! 10.5
 5!12.0!11.0! 4.0!16.0!19.0! 7.0 ! 11.5
 6!13.0! 7.0!12.0!17.5! 2.0! 9.0 ! 10.0
 7!11.0!13.0!11.5!12.0! 5.5!10.0 ! 10.5
 8!10.0! 8.5!15.0! 6.0! 7.5!11.0 !  9.5
 9! 9.0! 9.5!12.0! 7.0! 9.0!13.0 ! 10.0
10! 8.0!10.0!14.0! 7.0!10.0!17.0 ! 11.0
```

2.10 Occupazione massima di 48 Kbyte di memoria RAM

```
]
]PRINT FRE(0)+65536
47100

]10 DIM A(9000)

]RUN

]PRINT FRE(0)+65536
67611

]CLEAR

]PRINT FRE(0)+65536
47087

]10 DIM A%(20000)

]RUN

]PRINT FRE(0)+65536
72612
```

2.11 Frequenza delle lettere in un testo

```
10  REM  FREQUENZA DELLE LETTERE
11  REM  IN UNA FRASE
15  CLEAR
20  INPUT PH$
22  DIM F(26)
23  FOR I = 1 TO 26:F(I) = 0: NEXT I
25  FOR J = 1 TO  LEN (PH$)
26 C$ =  MID$ (PH$,J,1)
27  FOR I = 65 TO 90
30  IF C$ =  CHR$ (I) THEN F(I - 64) = F(I - 64) + 1
35  NEXT I
40  NEXT J
50  FOR I = 65 TO 90
51  IF F(I - 64) > 0 THEN  PRINT  CHR$ (I),F(I - 64)
55  NEXT I

]RUN
?DOVE ANDIAMO/
A              2
D              2
E              1
I              1
M              1
N              1
O              2
V              1
```

3.1 Spostamento di un quadratino su una diagonale

```
5   GR
10   COLOR= 3: PLOT 0,0: GOSUB 100
20   FOR X = 1 TO 39
30   COLOR= 0: PLOT X - 1,X - 1
40   COLOR= 3: PLOT X,X: GOSUB 100
50   NEXT X
60   COLOR= 0: PLOT 39,39
70   GOTO 10
100   FOR Z = 1 TO 400: NEXT Z
105   RETURN
```

3.2 Rimbalzo in un quadrato

```
5  HOME
10  REM  RIMBALZO
11  GR : COLOR= 4
12  REM  QUADRATO
13  HLIN 0,34 AT 0
14  VLIN 0,39 AT 0
15  HLIN 0,34 AT 39
16  VLIN 0,39 AT 34
20 A = 1:B = 2
24 X = 1:Y = 1
30 XP = X + A:YP = Y + B
40  IF XP > 33 THEN XP = 33:A =  - A: GOSUB 100
50  IF XP < 1 THEN XP = 1:A =  - A: GOSUB 100
60  IF YP > 38 THEN YP = 38:B =  - B: GOSUB 100
70  IF YP < 1 THEN YP = 1:B =  - B: GOSUB 100
75  COLOR= 0: PLOT X,Y
80  COLOR= 4: PLOT XP,YP
85 X = XP:Y = YP
90 ART =  PEEK ( - 16384)
92  IF ART < 128 THEN 30
93  POKE  - 16368,0
95  TEXT : HOME : END
99 :
100  REM  SUONO
110  FOR K = 1 TO 10
120 S =  PEEK ( - 16336)
125  NEXT K
130  RETURN
```

3.3 Motivo simmetrico

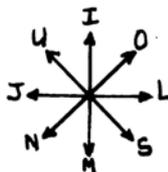
```
10  GR
20 X = 17:Y = 26
22 A =  - 1
25  COLOR= 7
27  IF Y <  = 1 OR Y > 27 THEN A =  - A
40  PLOT X - 2,Y + 6
50  PLOT X + 2,Y + 6
60  VLIN Y,Y + 1 AT X
70  COLOR= 9
80  PLOT X - 2,Y + 5
90  PLOT X + 2,Y + 5
100  PLOT X,Y + 2
110  COLOR= 15
```

```
120  VLIN Y + 3,Y + 4 AT X - 2
130  VLIN Y + 3,Y + 4 AT X + 2
140  VLIN Y + 7,Y + 8 AT X + 1
150  PLOT X,Y + 4
160  VLIN Y + 7,Y + 8 AT X - 1
170  COLOR= 4
180  HLIN X - 1,X + 1 AT Y + 5
190  HLIN X - 1,X + 1 AT Y + 6
200  HLIN X - 1,X + 1 AT Y + 3
210  VLIN Y + 9,Y + 11 AT X - 1
220  VLIN Y + 9,Y + 11 AT X + 1
230  PLOT X - 1,Y + 2
240  PLOT X + 1,Y + 2
250  COLOR= 11
260  PLOT X - 1,Y
270  PLOT X + 1,Y
280  PLOT X - 2,Y - 1
290  PLOT X + 2,Y - 1
300  GR :Y = Y + A: GOTO 25
```

3.4 Cubi

```
5  HGR : POKE  - 16302,0: HCOLOR= 3
10  REM  CUBI INCASTRATI
20  DIM X(8),Y(8)
25 X0 = 5:Y0 = 55
30  FOR C = 10 TO 100 STEP 10
40  REM  COORDINATE DELLE SOMME
41 X(1) = X0:Y(1) = Y0
42 X(2) = X0:Y(2) = Y0 + C
44 X(3) = X0 + C:Y(3) = Y0 + C
45 X(4) = X0 + C:Y(4) = Y0
46 X(5) = X0 + 1.5 * C:Y(5) = Y0 - .5 * C
47 X(6) = X0 + 1.5 * C:Y(6) = Y0 + .5 * C
48 X(7) = X0 + .5 * C:Y(7) = Y0 + .5 * C
49 X(8) = X0 + .5 * C:Y(8) = Y0 - .5 * C
50  GOSUB 10000
55  NEXT C
60  END
10000  REM  DISEGNO DEL CUBO
10005  HPLOT X(1),Y(1)
10010  FOR I = 2 TO 8
10020  HPLOT  TO X(I),Y(I)
10025  FOR K = 1 TO 500: NEXT K
10030  NEXT I
10040  HPLOT X(1),Y(1) TO X(8),Y(8)
10041  HPLOT X(2),Y(2) TO X(7),Y(7)
10042  HPLOT X(3),Y(3) TO X(6),Y(6)
10043  HPLOT X(5),Y(5) TO X(8),Y(8)
10044  HPLOT X(4),Y(4) TO X(1),Y(1)
10050  RETURN
```

3.5 Composizione di un disegno

```
10  REM  8 DIREZIONI
12  DIM DR$(7),MD(7,2)
14  FOR I = 0 TO 7
16  READ DR$(I)
18  FOR J = 0 TO 1
20  READ MD(I,J)
22  NEXT J,I
25  DATA  L,1,0,0,1,-1,I,0,-1,U,-1,-1,J,-1,0,N,-1,1,M,0,1,",",1,1
30  HGR
32  HCOLOR= 3
34  INPUT "PASSO ?";S: HPLOT 140,80
35 X = 140:Y = 80
36  GET A$: IF A$ =  CHR$ (13) THEN 100
37  FOR I = 0 TO 7
38  IF A$ = DR$(I) THEN 50
39  NEXT I
40  REM  COLORE
41  FOR I = 0 TO 7
42  IF A$ =  CHR$ (I + 48) THEN 45
43  NEXT I
44  HGR : GOTO 36
45  VTAB 24: PRINT "COLORE ";I;" "
46  HCOLOR= I: GOTO 36
47 :
49  REM  DISEGNO DEL PUNTO
50 X = X + S * MD(I,0):Y = Y + S * MD(I,1)
51  HPLOT X,Y
52 :
55  GOTO 36
100  TEXT : HOME : END
```

L. 10.000 Cod. 341 D ISBN88-7056-145-3

Per chi vuole meglio apprendere le immense risorse offerte dall'Apple, suggeriamo la lettura di questo libro che ha il preciso obiettivo di facilitare l'apprendimento del linguaggio BASIC APPLESOFT le cui normali istruzioni a volte appaiono misteriose. Dedicando un po' di tempo alla lettura, vi renderete conto del significato di certe istruzioni e riuscirete ad avvicinarvi al "vostro Apple" con uno spirito più scientifico ed analitico.
Il libro si compone di tre capitoli:
"*Il sistema APPLE II*" nel quale sono descritti accuratamente l'hardware e il software,
"*il BASIC APPLESOFT*" con le istruzioni, i sottoprogrammi, gli operatori aritmetici e logici, *il disegno e la grafica* con le zone di memorie RAM e le funzioni grafiche.
Tutti gli argomenti sono trattati con il preciso intento di far prendere coscienza delle reali possibilità dalla macchina (numeri reali, stringhe di caratteri, loro codifica interna, loro occupazione in memoria etc.).
Il tutto è arricchito da numerosi esempi ed esercizi con soluzione.

90
LA PRATICA DELL'APPLE
Nicole Bréaud
Pouliquen
GRUPPO
EDITORIALE
JACKSON